FROM EMOTION TO EMULATION

CELEBRATING 20 YEARS OF REVERSE ENGINEERING

whoami_

whoami_

GovanifY

whoami_

GovanifY

Gauvain Tanguy
Henri Gabriel
Isidore Roussel-Tarbouriech

whoami_

GovanifY

Gauvain Tanguy Henri Gabriel Isidore Roussel-Tarbouriech

- Core PCSX2 contributor

whoami_

GovanifY

Gauvain Tanguy Henri Gabriel Isidore Roussel-Tarbouriech

- Core PCSX2 contributor
- CS professor

whoami_

GovanifY

Gauvain Tanguy
Henri Gabriel
Isidore Roussel-Tarbouriech

- Core PCSX2 contributor
- CS professor
- VG industry contractor

whoami_

GovanifY

Gauvain Tanguy Henri Gabriel Isidore Roussel-Tarbouriech

- Core PCSX2 contributor
- CS professor
- VG industry contractor
- Reverse engineer, console hacker

whoarewe_

PCSX2

whoarewe_

PCSX2

A Sony PlayStation 2 Emulator

- 98.24% of playable games!

whoarewe_

PCSX2

A Sony PlayStation 2 Emulator

- 98.24% of playable games!
- Website's Alexa rank ~40k, most popular VG emulator I'm aware of

whoarewe_

PCSX2

A Sony PlayStation 2 Emulator

- 98.24% of playable games!
- Website's Alexa rank ~40k, most popular VG emulator I'm aware of
- Very complex software project

whoarewe_

PCSX2

A Sony PlayStation 2 Emulator

- 98.24% of playable games!
- Website's Alexa rank ~40k, most popular VG emulator I'm aware of
- Very complex software project
- 20 year old project


SONY
PlayStation 2


SONY
PlayStation 2
SONY
PlayStation
SELECT
START
ANALOG

TECHNICALLY A PS2?

THAT'S STRETCHING IT!

AND A BLACK BOX TO EMULATE

HARDWARE

BLACK BOX

HARDWARE

BLACK BOX

How do we break into one?

HARDWARE

BLACK BOX

How do we break into one?

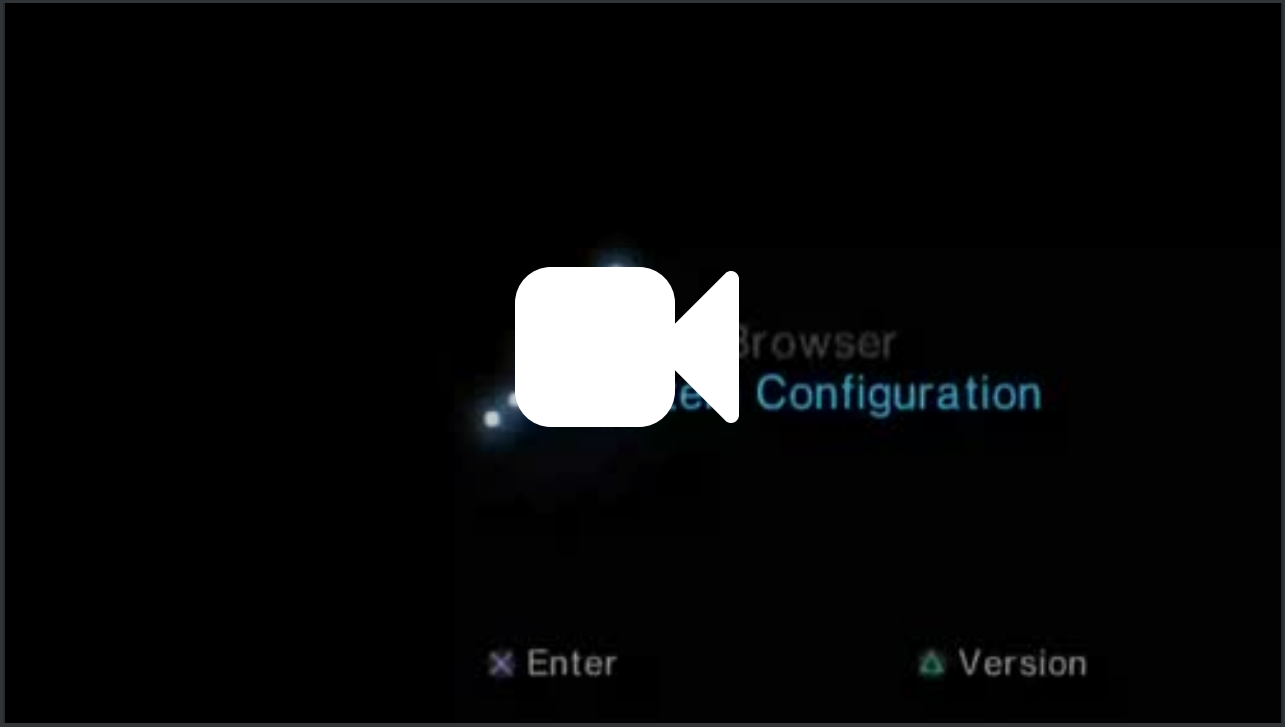
We hack its web browser!
- fail0verflow, circa 2013

HARDWARE

BLACK BOX

How do we break into one?

We hack its web browser!
- fail0verflow, circa 2013

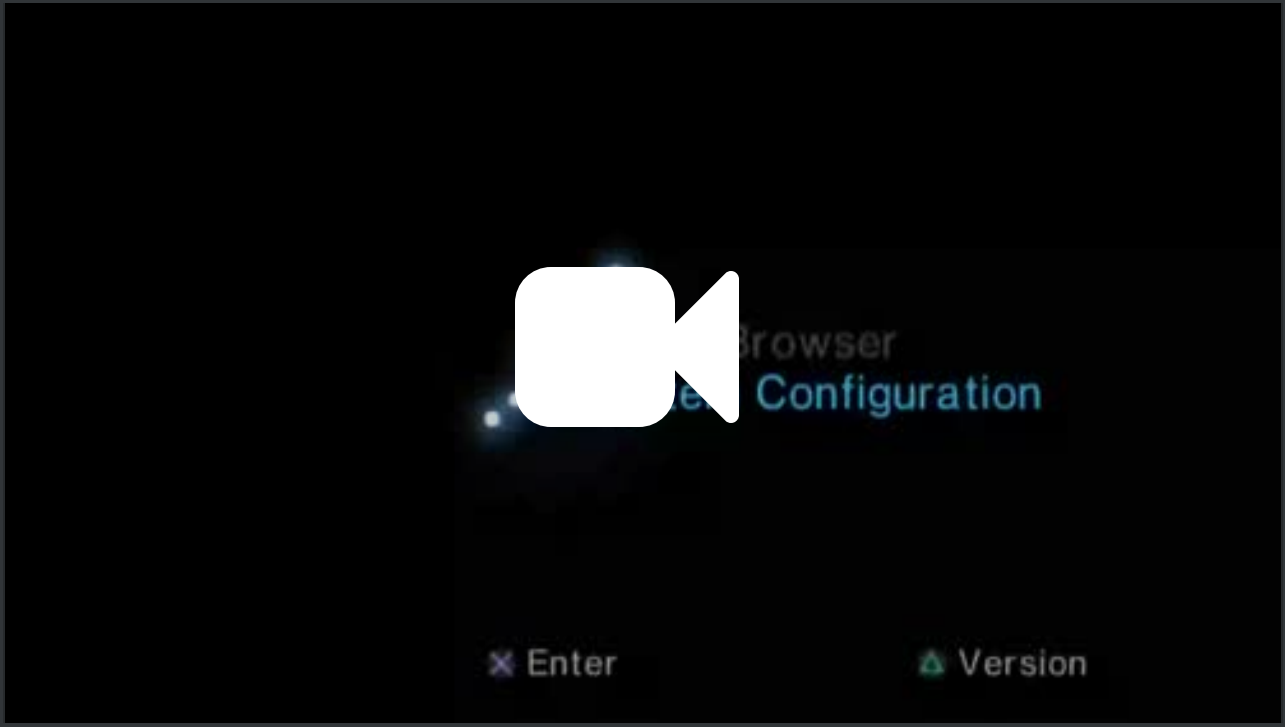


HARDWARE

BLACK BOX

How do we break into one?

We hack its web browser!
- fail0verflow, circa 2013



Us, circa 2002

HARDWARE

BLACK BOX

HARDWARE

BLACK BOX

Let's do it the good old way :)

HARDWARE

BLACK BOX

Let's do it the good old way :)

HARDWARE

BLACK BOX

Let's do it the good old way :)

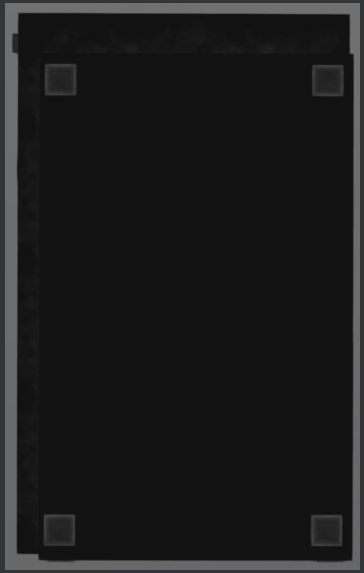
I'll explain first how all of this works and then how we figured it out


1
MAGICGATE
2
MEMORY CARD
MEMORY CARD
RESET
SONY

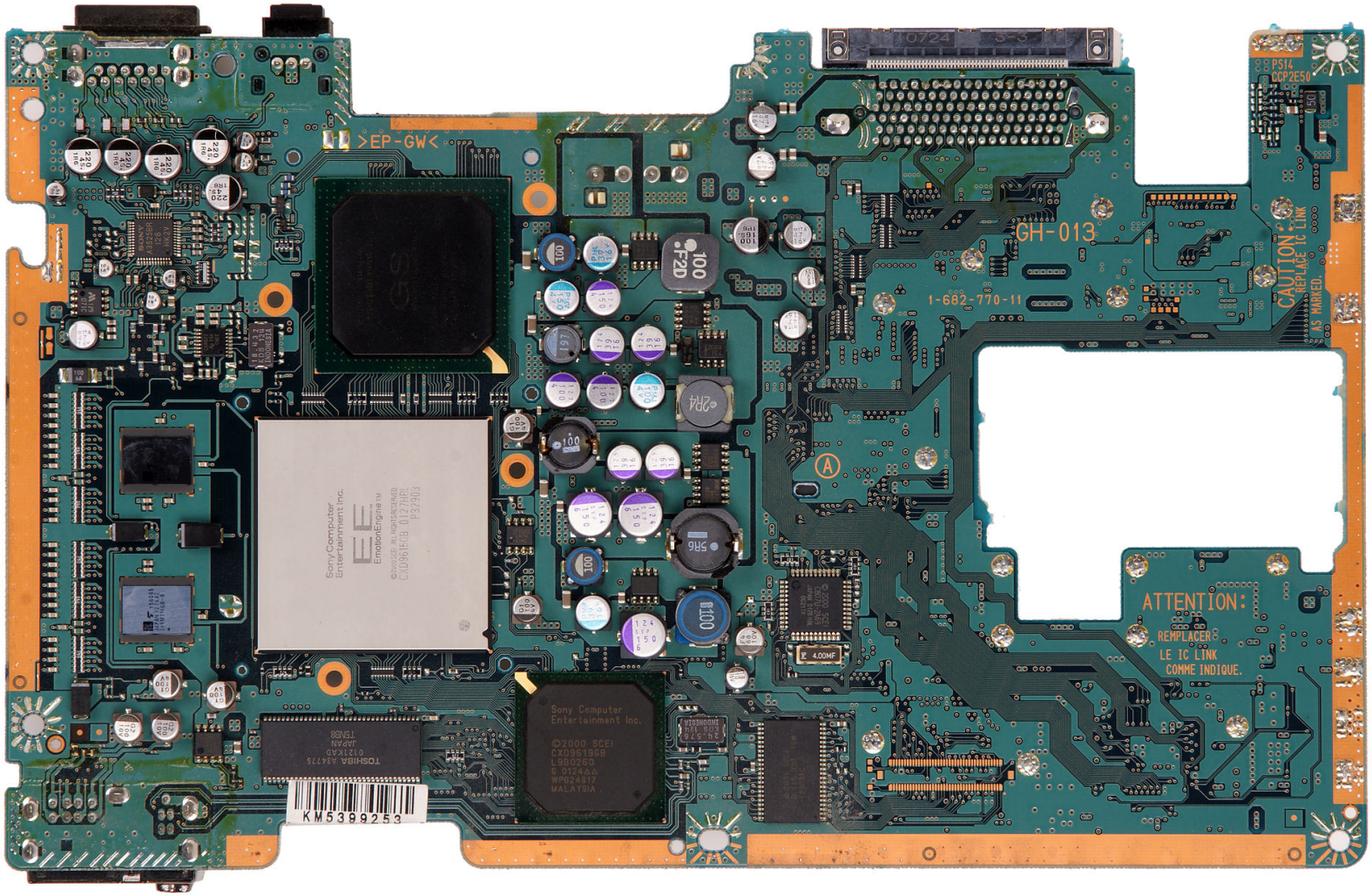
Reversible logo!


SONY
PlayStation 2
SCPH-70000
30-27206302-6188301
EXPANSION BAY

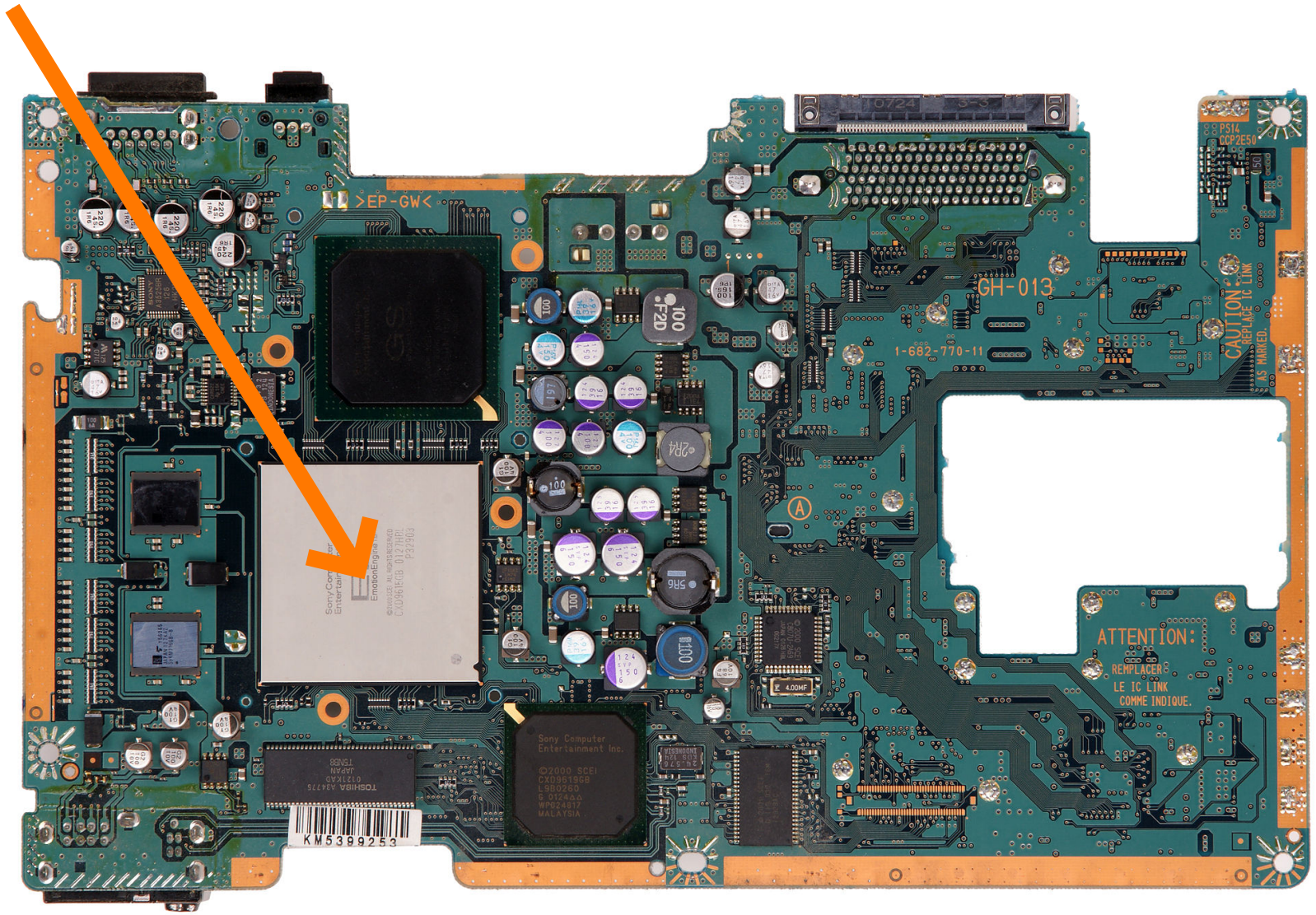

PS2
Playstation 2

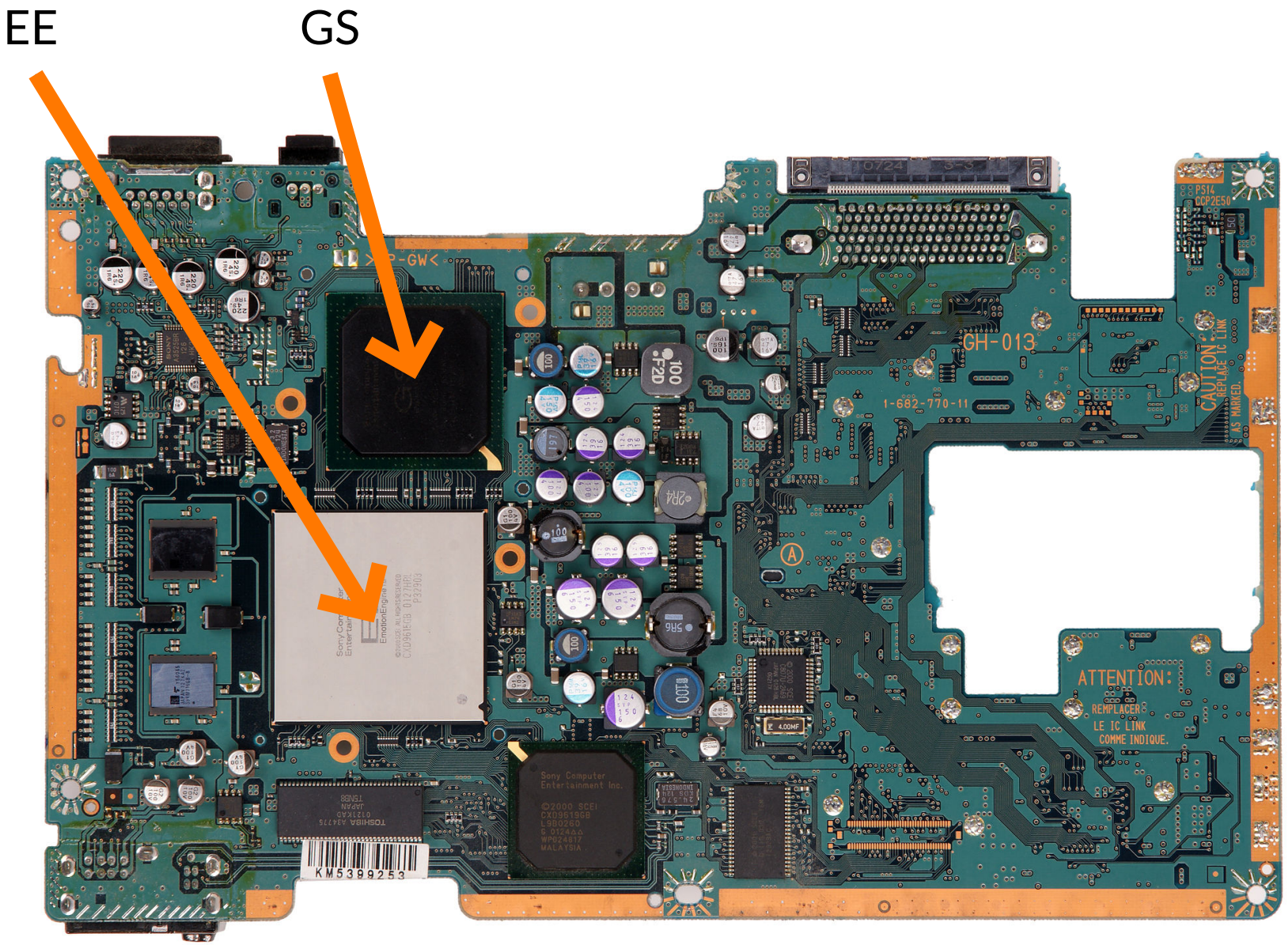


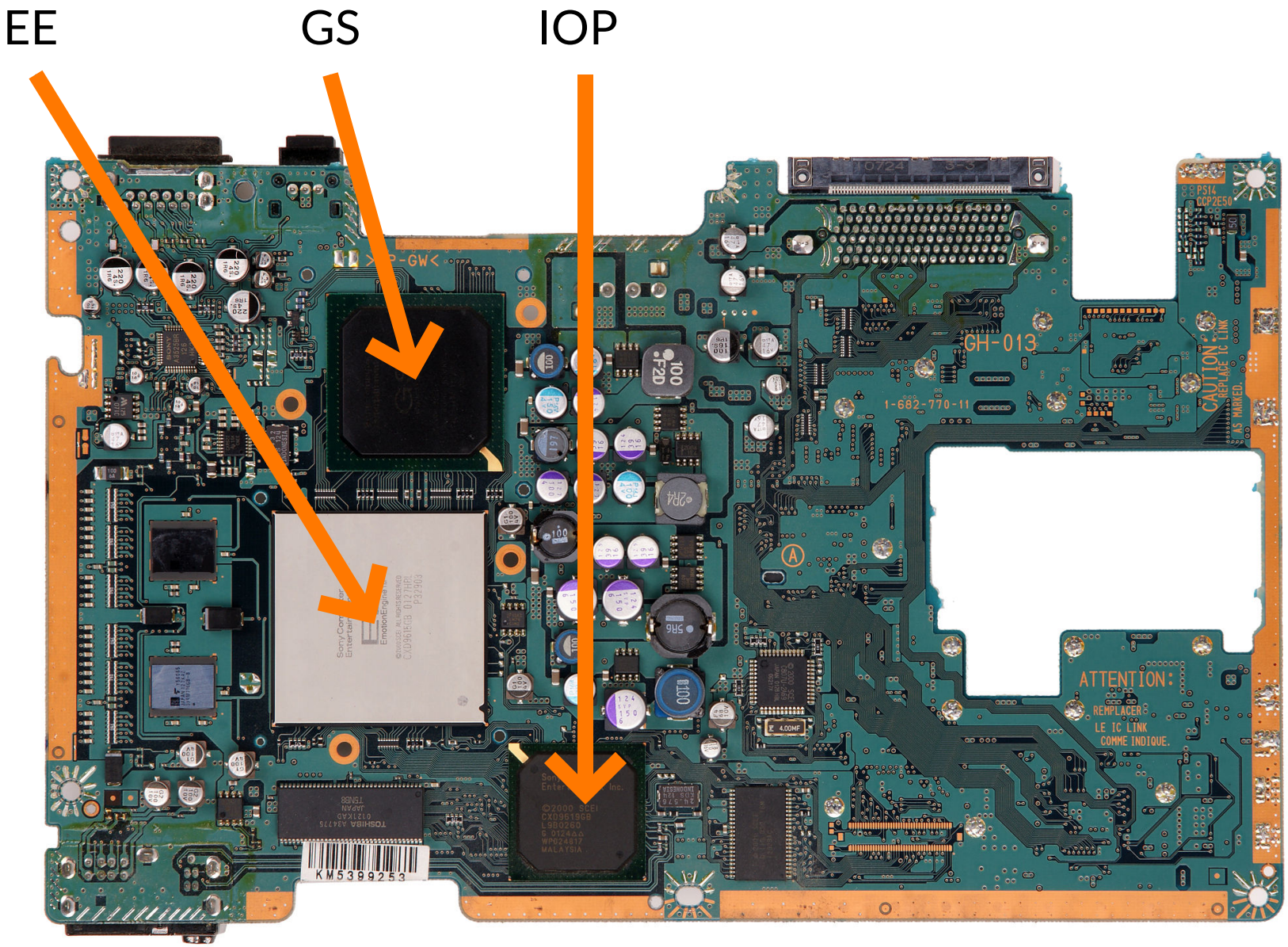

MAGICGATE
MEMORY CARD
MEMORY CARD
SONY

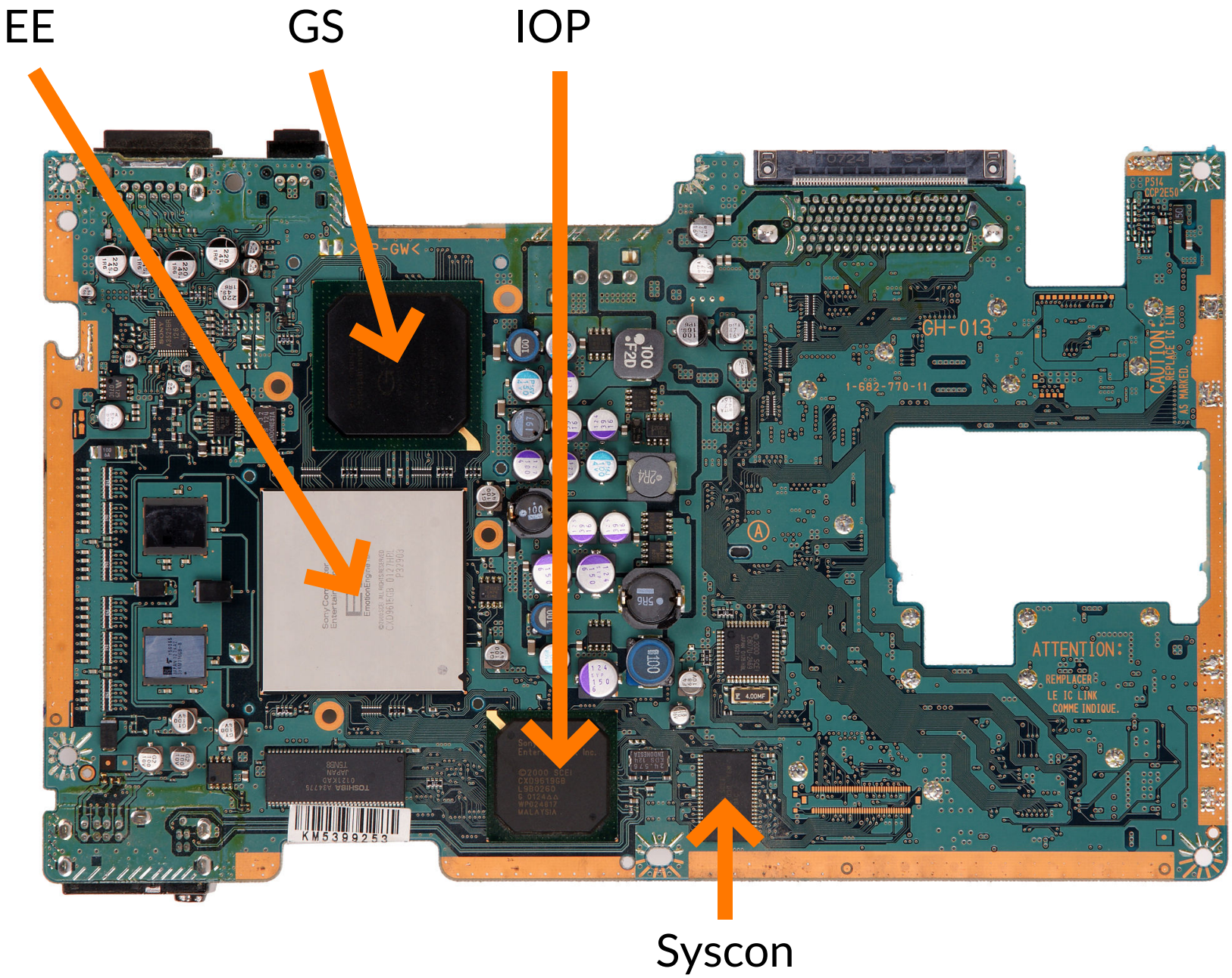

>EP-GW<
GH-013
1-682-770-11
CAUTION:
REPLACE IC LINK
AS MARKED.
ATTENTION:
REMPLACER
LE IC LINK
COMME INDIQUE.
Sony Computer
Entertainment Inc.
EE
EmotionEngine
TOSHIBA
JAPAN
KM5399253
Sony Computer
Entertainment Inc.
MALAYSIA
4.00MF

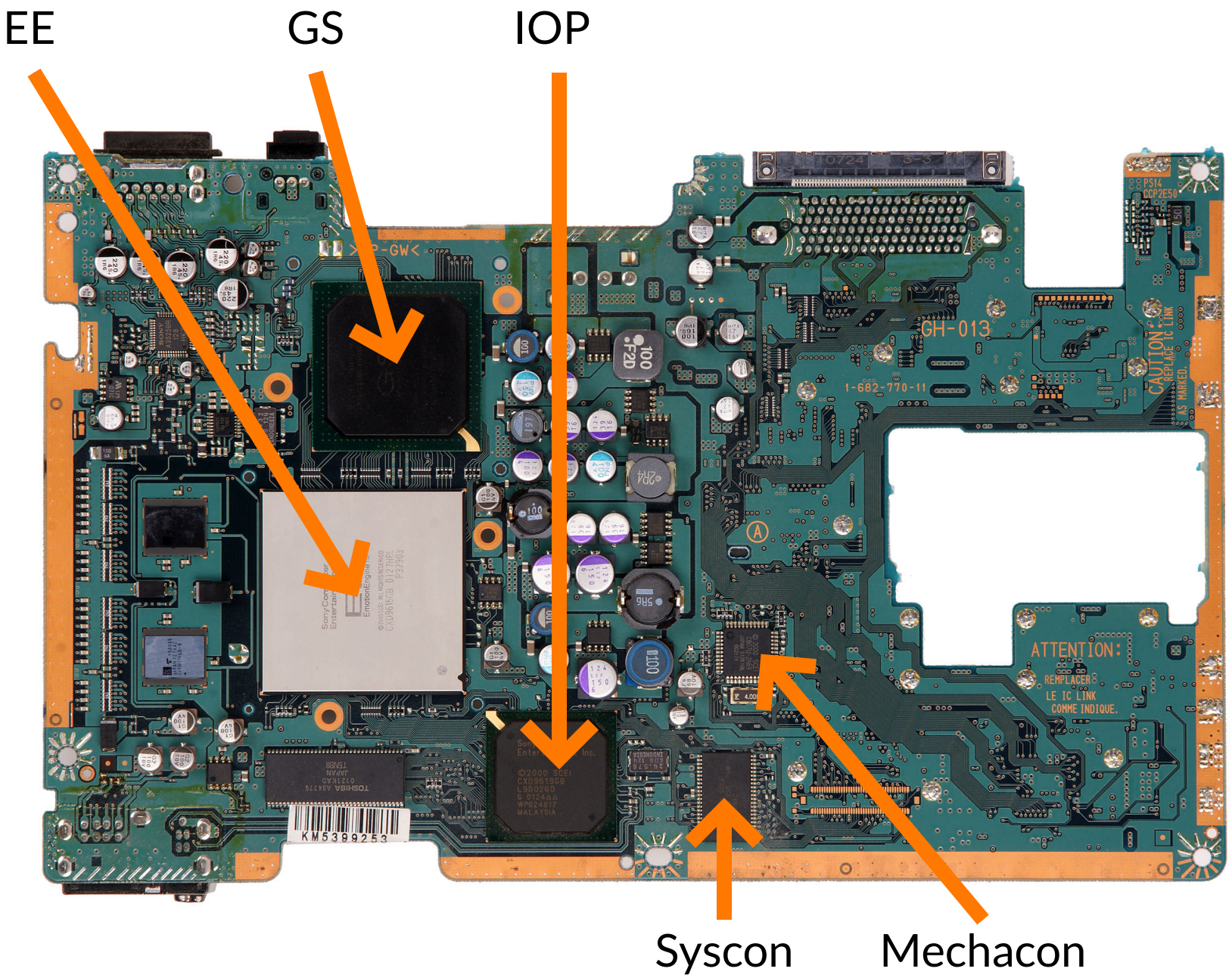
EE

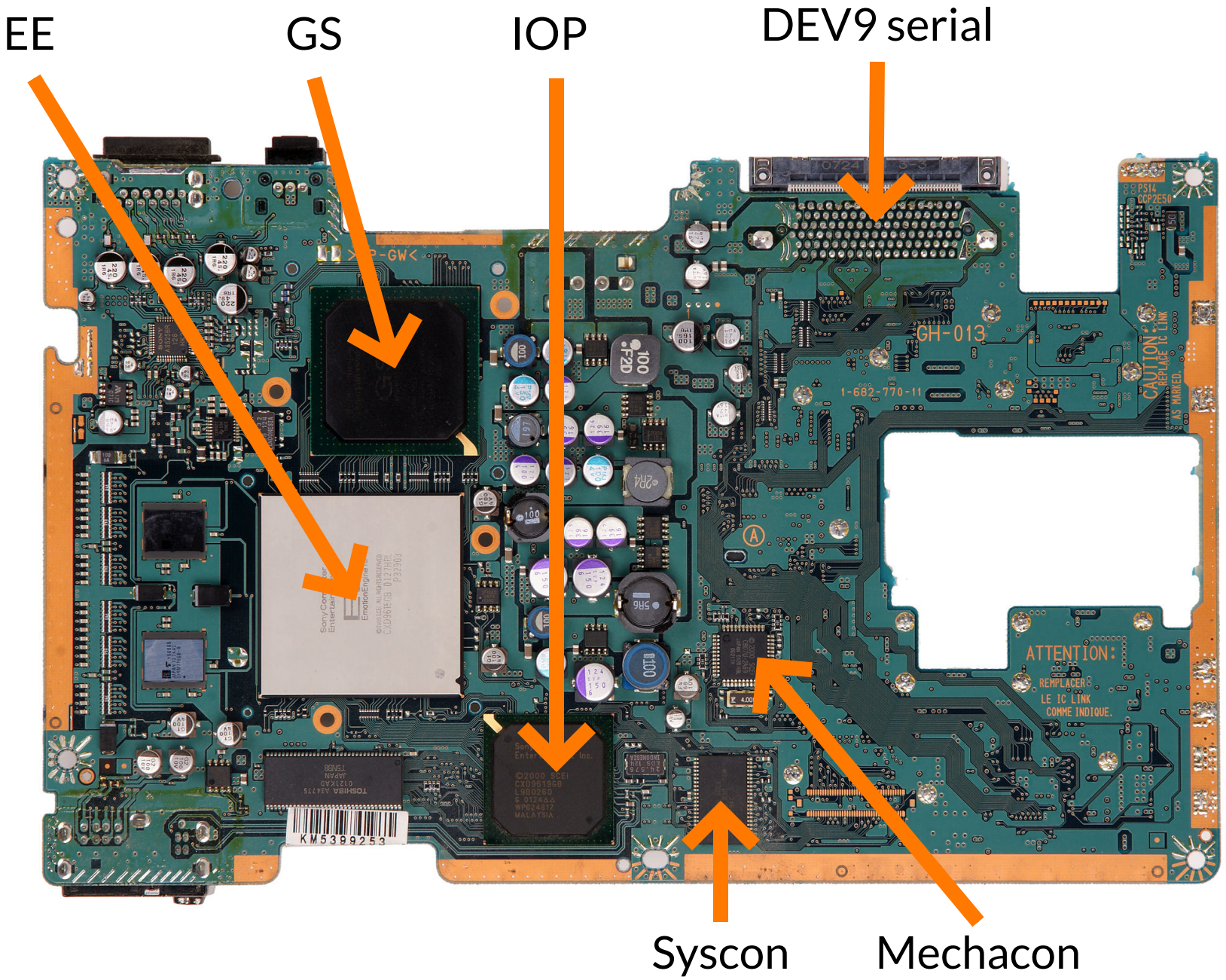

>EP-GW<
GH-013
1-682-770-11
CAUTION:
REPLACE IC LINK
AS MARKED.
ATTENTION:
REMPLACER
LE IC LINK
COMME INDIQUE.
Sony Computer
Entertainment Inc.
TOSHIBA
JAPAN
KM5399253

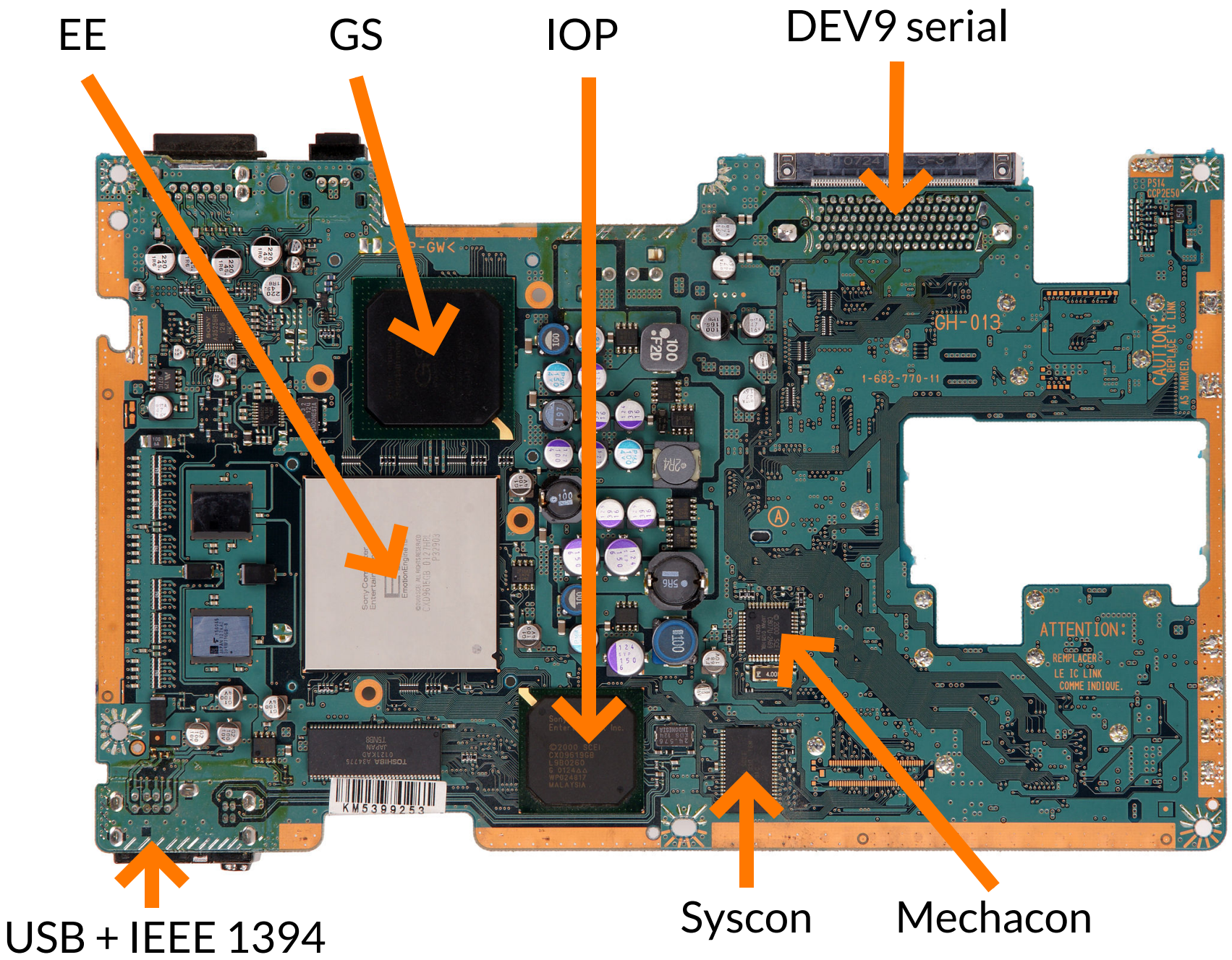

EE
GS
GH-013
1-682-770-11
ATTENTION:
REMPLACER
LE IC LINK
COMME INDIQUE.
CAUTION:
REPLACE IC LINK
AS MARKED.
KM5399253


EE
GS
IOP
GH-013
1-682-770-11
ATTENTION:
REMPLACER
LE IC LINK
COMME INDIQUE.
CAUTION:
REPLACE IC LINK
AS MARKED.
KM5399253

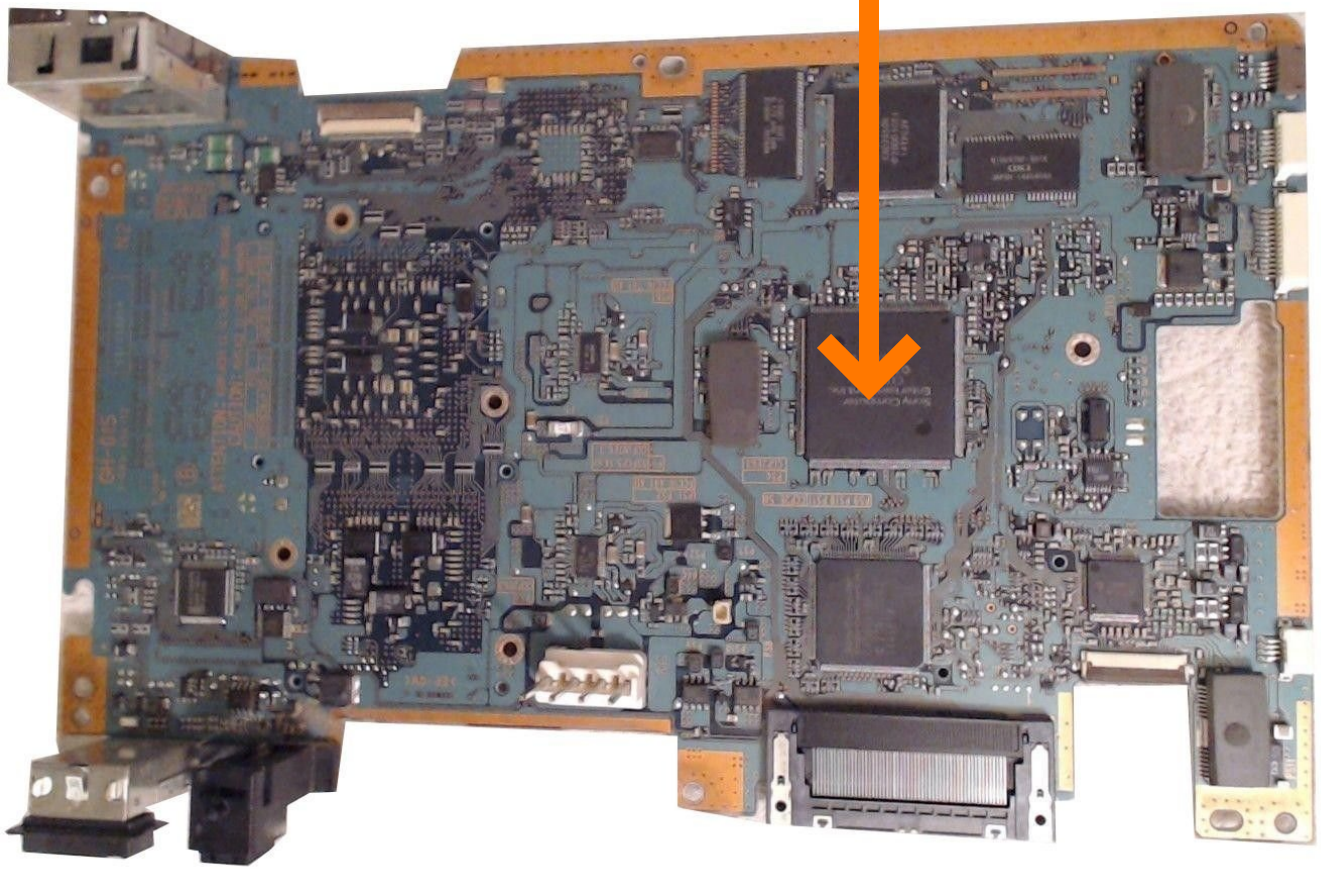

EE
GS
IOP
Syscon
GH-013
1-682-770-11
ATTENTION:
REMPLACER
LE IC LINK
COMME INDIQUE.
CAUTION:
REPLACE IC LINK
AS MARKED.
KM5399253

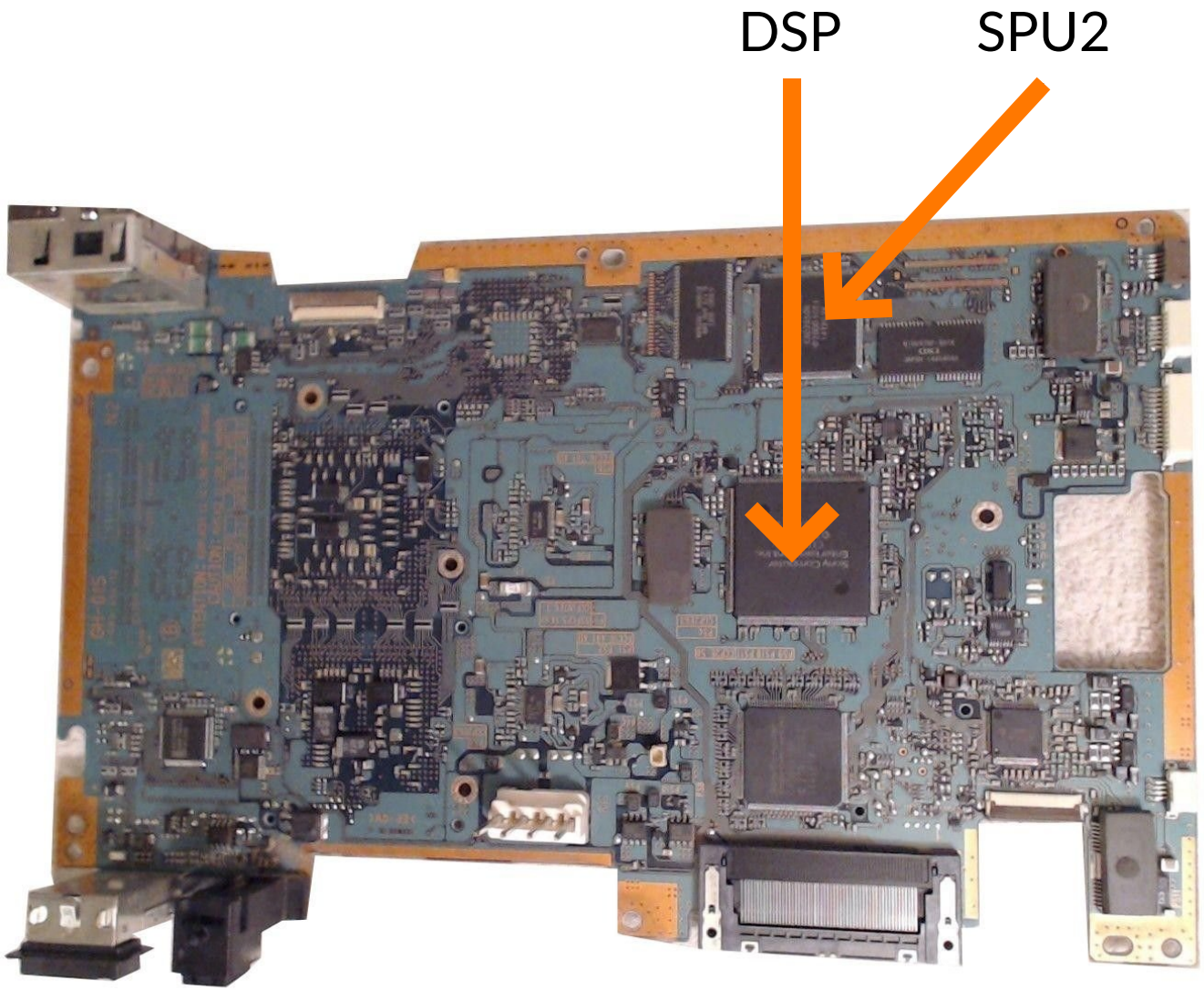

EE
GS
IOP
Syscon
Mechacon
GH-013
1-682-770-11
ATTENTION:
REMPLACER
LE IC LINK
COMME INDIQUE.
KM5399253

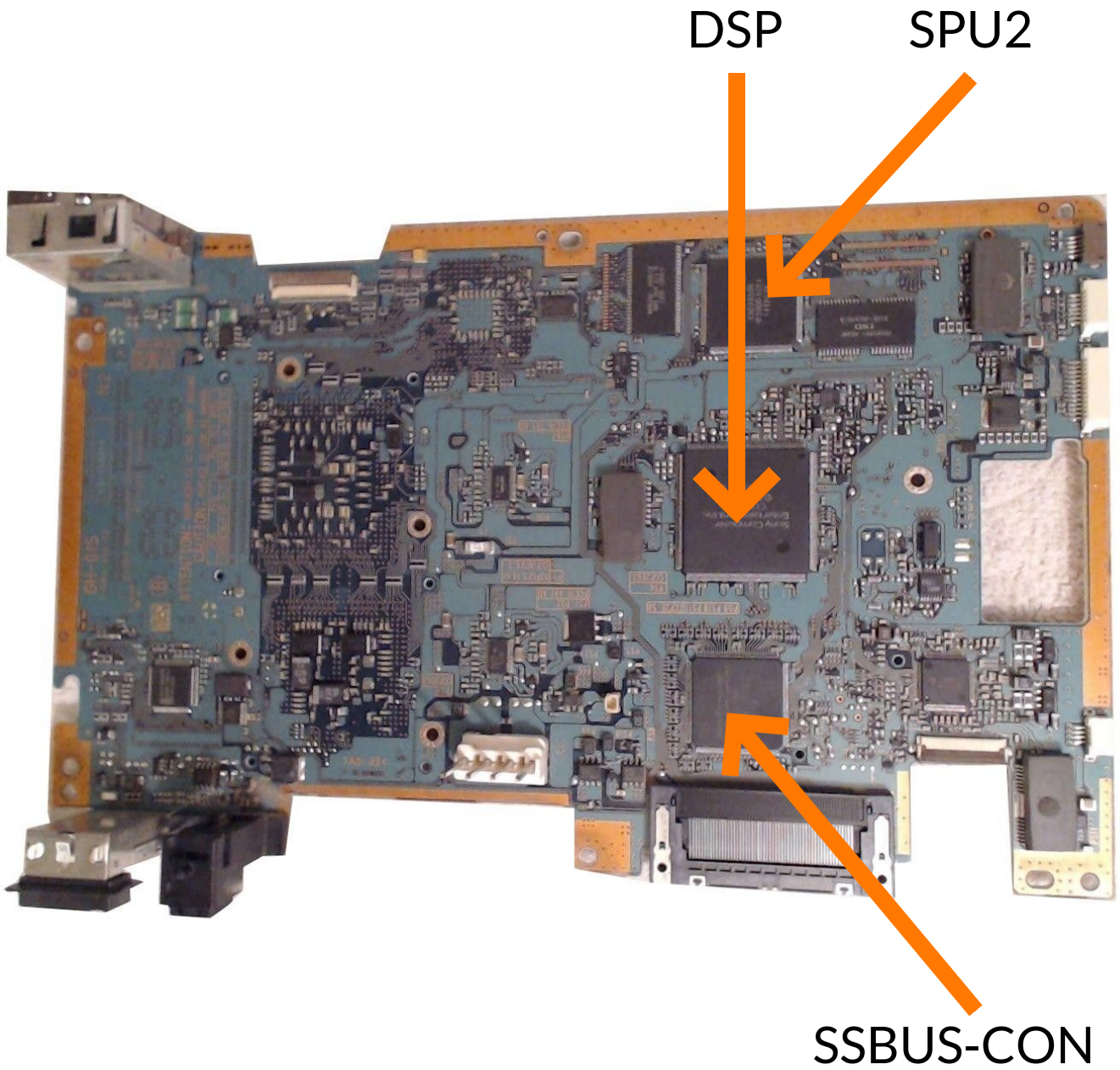

EE
GS
IOP
DEV9 serial
Syscon
Mechacon
GH-013
1-682-770-11
ATTENTION:
REMPLACER
LE IC LINK
COMME INDIQUE.
KM5399253

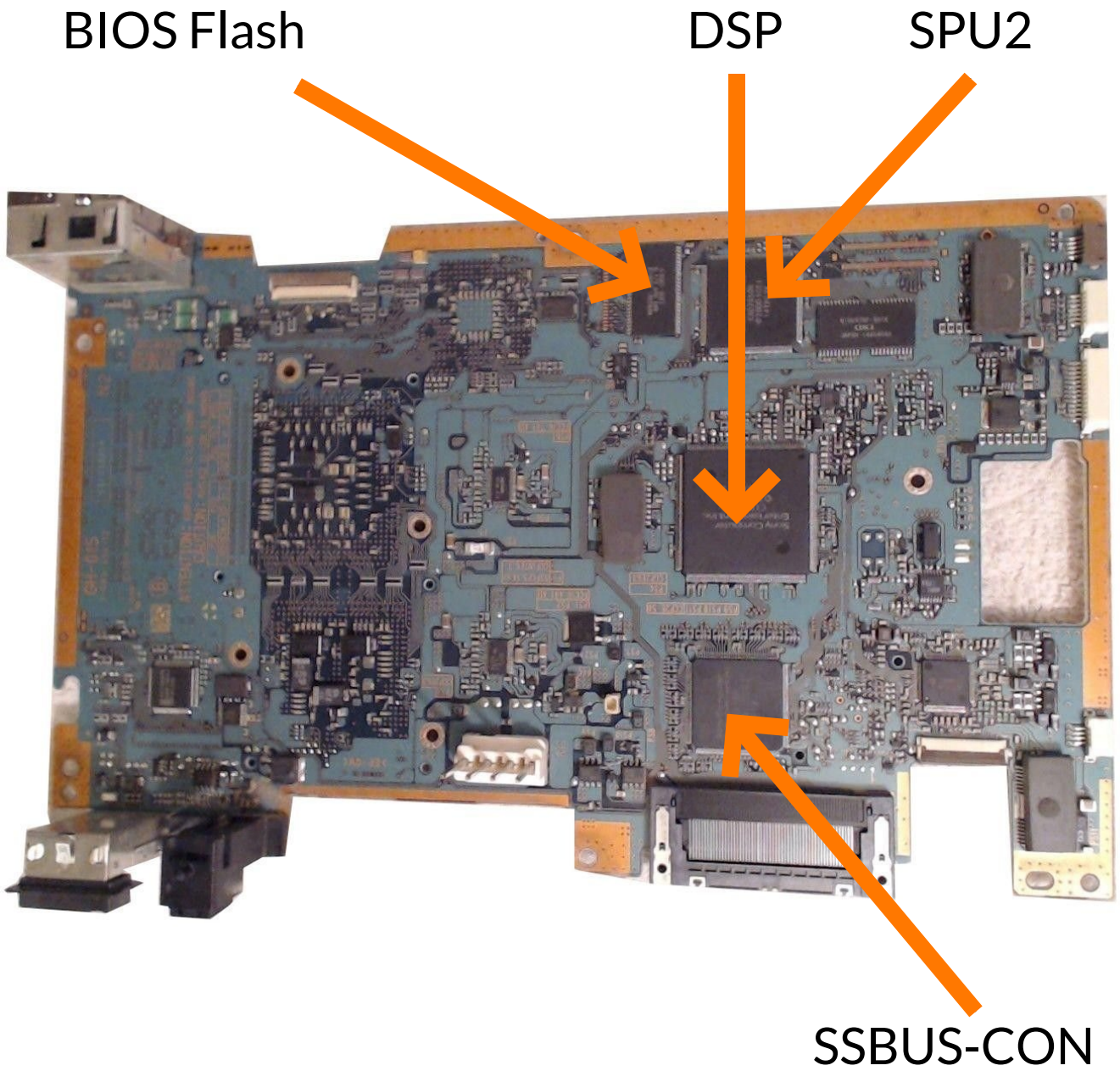

EE
GS
IOP
DEV9 serial
USB + IEEE 1394
Syscon
Mechacon
GH-013
1-682-770-11
ATTENTION:
REMPLACER
LE IC LINK
COMME INDIQUE.
KM5399253

DSP




DSP
SPU2


DSP
SPU2
SSBUS-CON


BIOS Flash
DSP
SPU2
SSBUS-CON

EE

WHAT IS THE EE

EE

WHAT IS THE EE

More like what contains the EE

EE

WHAT IS THE EE

More like what contains the EE

A core with 3 co-processors: (COP)

EE

WHAT IS THE EE

More like what contains the EE

A core with 3 co-processors: (COP)

- COP0: System co-processor

WHAT IS THE EE

More like what contains the EE

A core with 3 co-processors: (COP)

- COP0: System co-processor
- COP1: A floating point unit (FPU)

WHAT IS THE EE

More like what contains the EE

A core with 3 co-processors: (COP)

- COP0: System co-processor
- COP1: A floating point unit (FPU)
- COP2: Vector Unit 0 Macro Mode (VU0 - Macro)

WHAT IS THE EE

More like what contains the EE

A core with 3 co-processors: (COP)

- COP0: System co-processor
- COP1: A floating point unit (FPU)
- COP2: Vector Unit 0 Macro Mode (VU0 - Macro)

Also partly designed by a chip designer called coolchips for his Master's thesis, pretty cool!

EE

WHAT IS THE EE

EE

WHAT IS THE EE

- Image Processing Unit (IPU)

EE

WHAT IS THE EE

- Image Processing Unit (IPU)
- VPU1

EE

WHAT IS THE EE

- Image Processing Unit (IPU)
- VPU1
- VPU0 with VU0 accessible as a COP

EE

WHAT IS THE EE

- Image Processing Unit (IPU)
- VPU1
- VPU0 with VU0 accessible as a COP
- GIF/VIF (Graphics/Vector Interface)

EE

WHAT IS THE EE

- Image Processing Unit (IPU)
- VPU1
- VPU0 with VU0 accessible as a COP
- GIF/VIF (Graphics/Vector Interface)
- i/dCache + ScratchPad (on die memory)

EE

WHAT IS THE EE

- Image Processing Unit (IPU)
- VPU1
- VPU0 with VU0 accessible as a COP
- GIF/VIF (Graphics/Vector Interface)
- i/dCache + ScratchPad (on die memory)
- DMA Controller

EE

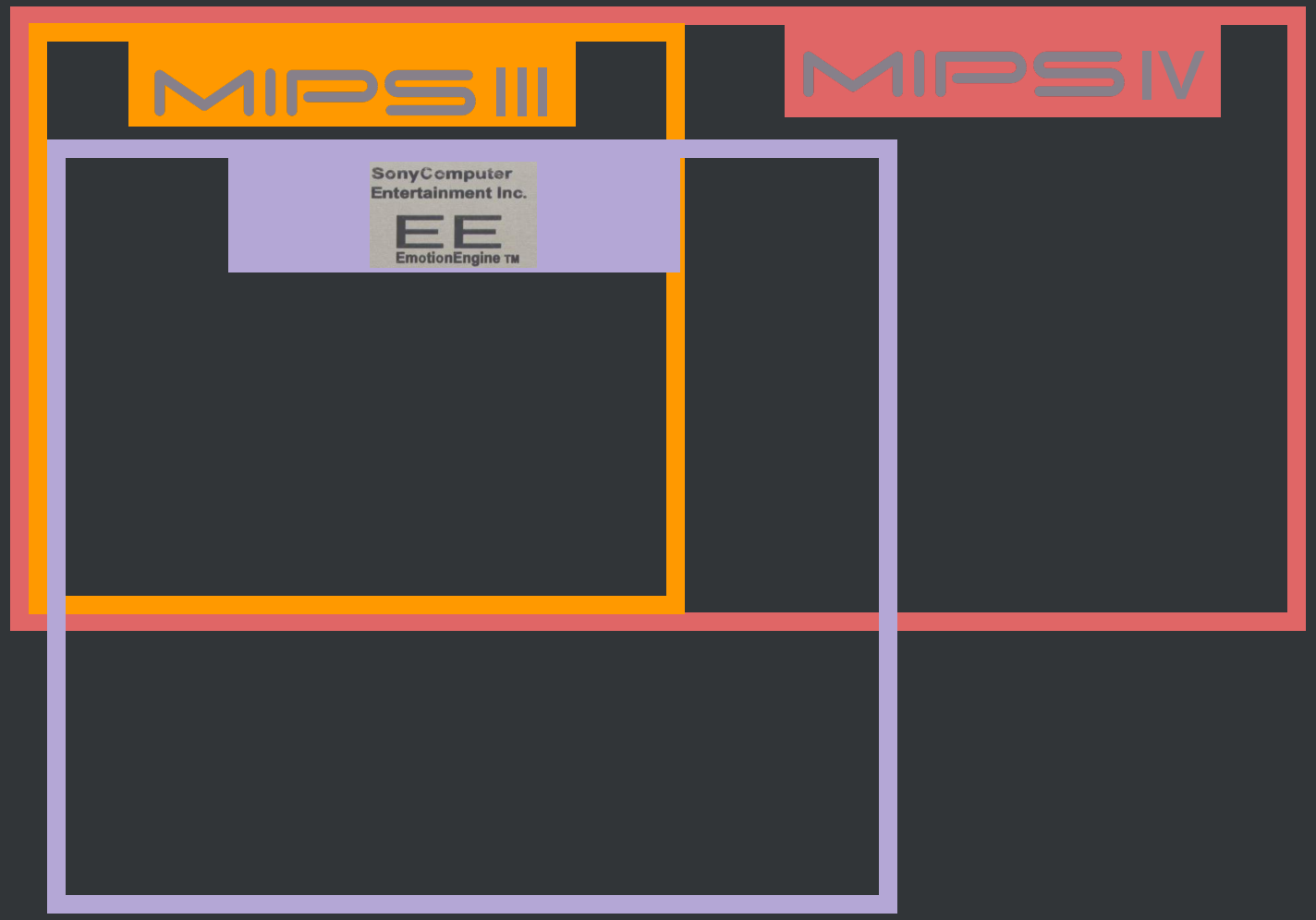
WHAT IS THE EE

- Image Processing Unit (IPU)
- VPU1
- VPU0 with VU0 accessible as a COP
- GIF/VIF (Graphics/Vector Interface)
- i/dCache + ScratchPad (on die memory)
- DMA Controller

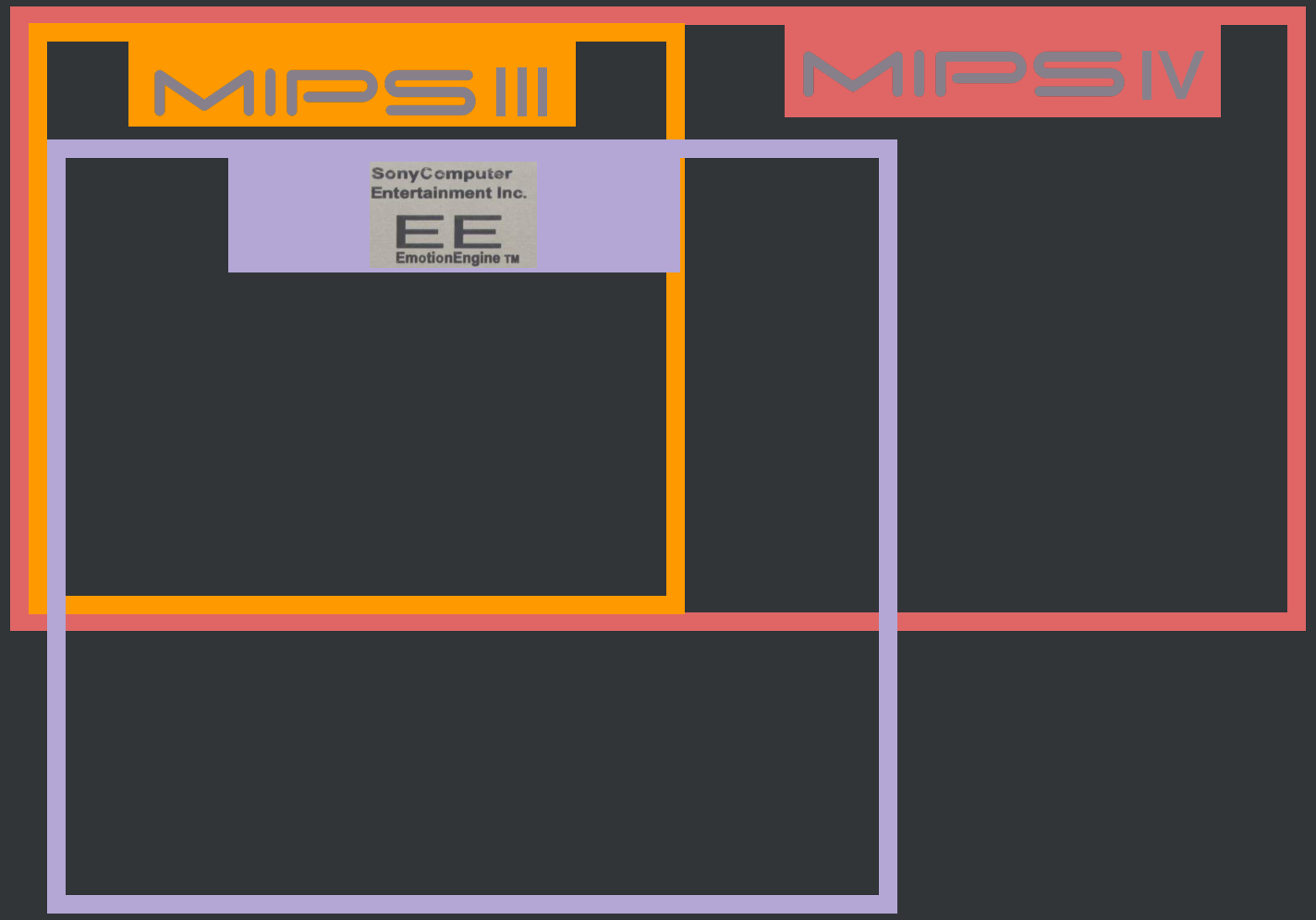
We're only getting started!

MIPS III


MIPS III
MIPS IV


MIPS III
MIPS IV
SonyComputer
Entertainment Inc.
EE
EmotionEngine TM

Only the best of MIPS have been used as we will see



EE Core - MIPS

```
1 ADD
2 ADDI
3 ADDIU
4 ADDU
5 AND
6 ANDI
7 BEQ
8 BEQL
9 BGEZ
10 BGEZAL
11 BGEZALL
12 BGEZL
13 BGTZ
14 BGTZL
15 BLEZ
16 BLEZL
17 BLTZ
18 BLTZAL
19 BLTZALL
20 BLTZL
21 BNE
22 BNEL
23 BREAK
24 DADD
25 DADDI
26 DADDIU
27 DADDU
28 DIV
29 DIVU
30 DSLL
31 DSLL32
32 DSLLV
33 DSRA
34 DSRA32
35 DSRAV
36 DSRL
37 DSRL32
38 DSRLV
39 DSUB
```

```
40 DSUBU
41 JAL
42 JALR
43 JR
44 LB
45 LBU
46 LD
47 LDL
48 LDR
49 LH
50 LHU
51 LUI
52 LW
53 LWL
54 LWR
55 LWU
56 MFHI
57 MFLO
58 MOVN
59 MOVZ
60 MTHI
61 MTLO
62 MULT
63 MULTU
64 NOR
65 OR
66 ORI
67 PREF
68 SB
69 SD
70 SDL
71 SDR
72 SH
73 SLL
74 SLLV
75 SLT
76 SLTI
77 SLTIU
78 SLTU
```

```
79 SRA
80 SRAV
81 SRL
82 SRLV
83 SUB
84 SUBU
85 SW
86 SWL
87 SWR
88 SYNC.stype
89 SYSCALL
90 TEQ
91 TEQI
92 TGE
93 TGEI
94 TGEIU
95 TGEU
96 TLT
97 TLTI
98 TLTIU
99 TLTU
100 TNE
101 TNEI
102 XOR
103 XORI
```

EE Core additions

```
1 ADD
2 ADDI
3 ADDIU
4 ADDU
5 AND
6 ANDI
7 BEQ
8 BEQL
9 BGEZ
10 BGEZAL
11 BGEZALL
12 BGEZL
13 BGTZ
14 BGTZL
15 BLEZ
16 BLEZL
17 BLTZ
18 BLTZAL
19 BLTZALL
20 BLTZL
21 BNE
22 BNEL
23 BREAK
24 DADD
25 DADDI
26 DADDIU
27 DADDU
28 DIV
29 DIVU
30 DSLL
31 DSLL32
32 DSLLV
33 DSRA
34 DSRA32
35 DSRAV
36 DSRL
37 DSRL32
38 DSRLV
39 DSUB
```

```
40 DSUBU
41 JAL
42 JALR
43 JR
44 LB
45 LBU
46 LD
47 LDL
48 LDR
49 LH
50 LHU
51 LUI
52 LW
53 LWL
54 LWR
55 LWU
56 MFHI
57 MFLO
58 MOVN
59 MOVZ
60 MTHI
61 MTLO
62 MULT
63 MULTU
64 NOR
65 OR
66 ORI
67 PREF
68 SB
69 SD
70 SDL
71 SDR
72 SH
73 SLL
74 SLLV
75 SLT
76 SLTI
77 SLTIU
78 SLTU
```

```
79 SRA
80 SRAV
81 SRL
82 SRLV
83 SUB
84 SUBU
85 SW
86 SWL
87 SWR
88 SYNC.stype
89 SYSCALL
90 TEQ
91 TEQI
92 TGE
93 TGEI
94 TGEIU
95 TGEU
96 TLT
97 TLTI
98 TLTIU
99 TLTU
100 TNE
101 TNEI
102 XOR
103 XORI
```

```
104 DIV1
105 DIVU1
106 LQ
107 MADD
108 MADD1
109 MADDU
110 MADDU1
111 MFHI1
112 MFLO1
113 MFSA
114 MTHI1
115 MTLO1
116 MTSA
117 MTSAB
118 MTSAH
119 MULT
120 MULT1
121 MULTU
122 MULTU1
123 PABSH
124 PABSW
125 PADDB
126 PADDH
127 PADDSB
128 PADDSH
129 PADDSW
130 PADDUB
131 PADDUH
132 PADDUW
133 PADDW
134 PADSBH
135 PAND
136 PCEQB
137 PCEQH
138 PCEQW
139 PCGTB
140 PCGTH
141 PCGTW
142 PCPYH
```

```
143 PCPYLD
144 PCPYUD
145 PDIVBW
146 PDIVUW
147 PDIVW
148 PEXCH
149 PEXCW
150 PEXEH
151 PEXEW
152 PEXT5
153 PEXTLB
154 PEXTLH
155 PEXTLW
156 PEXTUB
157 PEXTUH
158 PEXTUW
159 PHMADH
160 PHMSBH
161 PINTEH
162 PINTH
163 PLZCW
164 PMADDH
165 PMADDUW
166 PMADDW
167 PMAXH
168 PMAXW
169 PMFHI
170 PMFHL.LH
171 PMFHL.LW
172 PMFHL.SH
173 PMFHL.SLW
174 PMFHL.UW
175 PMFLO
176 PMINH
177 PMINW
178 PMSUBH
179 PMSUBW
180 PMTHI
181 PMTHL.LW
```

```
182 PMTLO
183 PMULTH
184 PMULTUW
185 PMULTW
186 PNOR
187 POR
188 PPAC5
189 PPACB
190 PPACH
191 PPACW
192 PREVH
193 PROT3W
194 PSLLH
195 PSLLVW
196 PSLLW
197 PSRAH
198 PSRAVW
199 PSRAW
200 PSRLH
201 PSRLVW
202 PSRLW
203 PSUBB
204 PSUBH
205 PSUBSB
206 PSUBSH
207 PSUBSW
208 PSUBUB
209 PSUBUH
210 PSUBUW
211 PSUBW
212 PXOR
213 QFSRV
214 SQ
```

EE - COP0

```
1 ADD
2 ADDI
3 ADDIU
4 ADDU
5 AND
6 ANDI
7 BEQ
8 BEQL
9 BGEZ
10 BGEZAL
11 BGEZALL
12 BGEZL
13 BGTZ
14 BGTZL
15 BLEZ
16 BLEZL
17 BLTZ
18 BLTZAL
19 BLTZALL
20 BLTZL
21 BNE
22 BNEL
23 BREAK
24 DADD
25 DADDI
26 DADDIU
27 DADDU
28 DIV
29 DIVU
30 DSLL
31 DSLL32
32 DSLLV
33 DSRA
34 DSRA32
35 DSRAV
36 DSRL
37 DSRL32
38 DSRLV
39 DSUB
```

```
40 DSUBU
41 JAL
42 JALR
43 JR
44 LB
45 LBU
46 LD
47 LDL
48 LDR
49 LH
50 LHU
51 LUI
52 LW
53 LWL
54 LWR
55 LWU
56 MFHI
57 MFLO
58 MOVN
59 MOVZ
60 MTHI
61 MTLO
62 MULT
63 MULTU
64 NOR
65 OR
66 ORI
67 PREF
68 SB
69 SD
70 SDL
71 SDR
72 SH
73 SLL
74 SLLV
75 SLT
76 SLTI
77 SLTIU
78 SLTU
```

```
79 SRA
80 SRAV
81 SRL
82 SRLV
83 SUB
84 SUBU
85 SW
86 SWL
87 SWR
88 SYNC.stype
89 SYSCALL
90 TEQ
91 TEQI
92 TGE
93 TGEI
94 TGEIU
95 TGEU
96 TLT
97 TLTI
98 TLTIU
99 TLTU
100 TNE
101 TNEI
102 XOR
103 XORI
```

```
104 DIV1
105 DIVU1
106 LQ
107 MADD
108 MADD1
109 MADDU
110 MADDU1
111 MFHI1
112 MFLO1
113 MFSA
114 MTHI1
115 MTLO1
116 MTSA
117 MTSAB
118 MTSAH
119 MULT
120 MULT1
121 MULTU
122 MULTU1
123 PABSH
124 PABSW
125 PADDB
126 PADDH
127 PADDSB
128 PADDSH
129 PADDSW
130 PADDUB
131 PADDUH
132 PADDUW
133 PADDW
134 PADSBH
135 PAND
136 PCEQB
137 PCEQH
138 PCEQW
139 PCGTB
140 PCGTH
141 PCGTW
142 PCPYH
```

```
143 PCPYLD
144 PCPYUD
145 PDIVBW
146 PDIVUW
147 PDIVW
148 PEXCH
149 PEXCW
150 PEXEH
151 PEXEW
152 PEXT5
153 PEXTLB
154 PEXTLH
155 PEXTLW
156 PEXTUB
157 PEXTUH
158 PEXTUW
159 PHMADH
160 PHMSBH
161 PINTEH
162 PINTH
163 PLZCW
164 PMADDH
165 PMADDUW
166 PMADDW
167 PMAXH
168 PMAXW
169 PMFHI
170 PMFHL.LH
171 PMFHL.LW
172 PMFHL.SH
173 PMFHL.SLW
174 PMFHL.UW
175 PMFLO
176 PMINH
177 PMINW
178 PMSUBH
179 PMSUBW
180 PMTHI
181 PMTHL.LW
```

```
182 PMTLO
183 PMULTH
184 PMULTUW
185 PMULTW
186 PNOR
187 POR
188 PPAC5
189 PPACB
190 PPACH
191 PPACW
192 PREVH
193 PROT3W
194 PSLLH
195 PSLLVW
196 PSLLW
197 PSRAH
198 PSRAVW
199 PSRAW
200 PSRLH
201 PSRLVW
202 PSRLW
203 PSUBB
204 PSUBH
205 PSUBSB
206 PSUBSH
207 PSUBSW
208 PSUBUB
209 PSUBUH
210 PSUBUW
211 PSUBW
212 PXOR
213 QFSRV
214 SQ
```

```
215 BC0F
216 BC0FL
217 BC0T
218 BC0TL
219 CACHE BFH
220 CACHE BHINBT
221 CACHE BXLBT
222 CACHE BXSBT
223 CACHE DHIN
224 CACHE DHWBIN
225 CACHE DHWOIN
226 CACHE DXIN
227 CACHE DXLDT
228 CACHE DXLTG
229 CACHE DXSDT
230 CACHE DXSTG
231 CACHE DXWBIN
232 CACHE IFL
233 CACHE IHIN
234 CACHE IXIN
235 CACHE IXLDT
236 CACHE IXLTG
237 CACHE IXSDT
238 CACHE IXSTG
239 DI
240 EI
241 ERET
242 MFBPC
243 MFC0
244 MFDAB
245 MFDABM
246 MFDVB
247 MFDVBM
248 MFIAB
249 MFIABM
250 MFPC
251 MFPS
252 MTBPC
253 MTC0
```

```
254 MTDAB
255 MTDABM
256 MTDVB
257 MTDVBM
258 MTIAB
259 MTIABM
260 MTPC
261 MTPS
262 TLBP
263 TLBR
264 TLBWI
265 TLBWR
```

EE - COP1

```
1 ADD
2 ADDI
3 ADDIU
4 ADDU
5 AND
6 ANDI
7 BEQ
8 BEQL
9 BGEZ
10 BGEZAL
11 BGEZALL
12 BGEZL
13 BGTZ
14 BGTZL
15 BLEZ
16 BLEZL
17 BLTZ
18 BLTZAL
19 BLTZALL
20 BLTZL
21 BNE
22 BNEL
23 BREAK
24 DADD
25 DADDI
26 DADDIU
27 DADDU
28 DIV
29 DIVU
30 DSLL
31 DSLL32
32 DSLLV
33 DSRA
34 DSRA32
35 DSRAV
36 DSRL
37 DSRL32
38 DSRLV
39 DSUB
```

```
40 DSUBU
41 JAL
42 JALR
43 JR
44 LB
45 LBU
46 LD
47 LDL
48 LDR
49 LH
50 LHU
51 LUI
52 LW
53 LWL
54 LWR
55 LWU
56 MFHI
57 MFLO
58 MOVN
59 MOVZ
60 MTHI
61 MTLO
62 MULT
63 MULTU
64 NOR
65 OR
66 ORI
67 PREF
68 SB
69 SD
70 SDL
71 SDR
72 SH
73 SLL
74 SLLV
75 SLT
76 SLTI
77 SLTIU
78 SLTU
```

```
79 SRA
80 SRAV
81 SRL
82 SRLV
83 SUB
84 SUBU
85 SW
86 SWL
87 SWR
88 SYNC.stype
89 SYSCALL
90 TEQ
91 TEQI
92 TGE
93 TGEI
94 TGEIU
95 TGEU
96 TLT
97 TLTI
98 TLTIU
99 TLTU
100 TNE
101 TNEI
102 XOR
103 XORI
```

```
104 DIV1
105 DIVU1
106 LQ
107 MADD
108 MADD1
109 MADDU
110 MADDU1
111 MFHI1
112 MFLO1
113 MFSA
114 MTHI1
115 MTLO1
116 MTSA
117 MTSAB
118 MTSAH
119 MULT
120 MULT1
121 MULTU
122 MULTU1
123 PABSH
124 PABSW
125 PADDB
126 PADDH
127 PADDSB
128 PADDSH
129 PADDSW
130 PADDUB
131 PADDUH
132 PADDUW
133 PADDW
134 PADSBH
135 PAND
136 PCEQB
137 PCEQH
138 PCEQW
139 PCGTB
140 PCGTH
141 PCGTW
142 PCPYH
```

```
143 PCPYLD
144 PCPYUD
145 PDIVBW
146 PDIVUW
147 PDIVW
148 PEXCH
149 PEXCW
150 PEXEH
151 PEXEW
152 PEXT5
153 PEXTLB
154 PEXTLH
155 PEXTLW
156 PEXTUB
157 PEXTUH
158 PEXTUW
159 PHMADH
160 PHMSBH
161 PINTEH
162 PINTH
163 PLZCW
164 PMADDH
165 PMADDUW
166 PMADDW
167 PMAXH
168 PMAXW
169 PMFHI
170 PMFHL.LH
171 PMFHL.LW
172 PMFHL.SH
173 PMFHL.SLW
174 PMFHL.UW
175 PMFLO
176 PMINH
177 PMINW
178 PMSUBH
179 PMSUBW
180 PMTHI
181 PMTHL.LW
```

```
182 PMTLO
183 PMULTH
184 PMULTUW
185 PMULTW
186 PNOR
187 POR
188 PPAC5
189 PPACB
190 PPACH
191 PPACW
192 PREVH
193 PROT3W
194 PSLLH
195 PSLLVW
196 PSLLW
197 PSRAH
198 PSRAVW
199 PSRAW
200 PSRLH
201 PSRLVW
202 PSRLW
203 PSUBB
204 PSUBH
205 PSUBSB
206 PSUBSH
207 PSUBSW
208 PSUBUB
209 PSUBUH
210 PSUBUW
211 PSUBW
212 PXOR
213 QFSRV
214 SQ
```

```
215 BC0F
216 BC0FL
217 BC0T
218 BC0TL
219 CACHE BFH
220 CACHE BHINBT
221 CACHE BXLBT
222 CACHE BXSBT
223 CACHE DHIN
224 CACHE DHWBIN
225 CACHE DHWOIN
226 CACHE DXIN
227 CACHE DXLDT
228 CACHE DXLTG
229 CACHE DXSDT
230 CACHE DXSTG
231 CACHE DXWBIN
232 CACHE IFL
233 CACHE IHIN
234 CACHE IXIN
235 CACHE IXLDT
236 CACHE IXLTG
237 CACHE IXSDT
238 CACHE IXSTG
239 DI
240 EI
241 ERET
242 MFBPC
243 MFC0
244 MFDAB
245 MFDABM
246 MFDVB
247 MFDVBM
248 MFIAB
249 MFIABM
250 MFPC
251 MFPS
252 MTBPC
253 MTC0
```

```
254 MTDAB
255 MTDABM
256 MTDVB
257 MTDVBM
258 MTIAB
259 MTIABM
260 MTPC
261 MTPS
262 TLBP
263 TLBR
264 TLBWI
265 TLBWR
```

```
266 ABS.S
267 ADD.S
268 ADD
269 ADDA.S
270 BC1F
271 BC1FL
272 FP False
273 BC1T
274 BC1TL
275 FP True Likely
276 CFC1
277 CTC1
278 CVT.S.W
279 CVT.W.S
280 DIV.S
281 LWC1
282 MADD.S
283 MADDA.S
284 MAX.S
285 MFC1
286 MIN.S
287 MOV.S
288 MSUB.S
289 MSUBA.S
290 MTC1
291 MUL.S
292 MULA.S
293 NEG.S
294 RSQRT.S
295 SQRT.S
296 SUB.S
297 SUBA.S
298 SWC1
```

EE - COP2

1 ADD
2 ADDI
3 ADDIU
4 ADDU
5 AND
6 ANDI
7 BEQ
8 BEQL
9 BGEZ
10 BGEZAL
11 BGEZALL
12 BGEZL
13 BGTZ
14 BGTZL
15 BLEZ
16 BLEZL
17 BLTZ
18 BLTZAL
19 BLTZALL
20 BLTZL
21 BNE
22 BNEL
23 BREAK
24 DADD
25 DADDI
26 DADDIU
27 DADDU
28 DIV
29 DIVU
30 DSLL
31 DSLL32
32 DSLLV
33 DSRA
34 DSRA32
35 DSRAV
36 DSRL
37 DSRL32
38 DSRLV
39 DSUB

40 DSUBU
41 JAL
42 JALR
43 JR
44 LB
45 LBU
46 LD
47 LDL
48 LDR
49 LH
50 LHU
51 LUI
52 LW
53 LWL
54 LWR
55 LWU
56 MFHI
57 MFLO
58 MOVN
59 MOVZ
60 MTHI
61 MTLO
62 MULT
63 MULTU
64 NOR
65 OR
66 ORI
67 PREF
68 SB
69 SD
70 SDL
71 SDR
72 SH
73 SLL
74 SLLV
75 SLT
76 SLTI
77 SLTIU
78 SLTU

79 SRA
80 SRAV
81 SRL
82 SRLV
83 SUB
84 SUBU
85 SW
86 SWL
87 SWR
88 SYNC.stype
89 SYSCALL
90 TEQ
91 TEQI
92 TGE
93 TGEI
94 TGEIU
95 TGEU
96 TLT
97 TLTI
98 TLTIU
99 TLTU
100 TNE
101 TNEI
102 XOR
103 XORI

104 DIV1
105 DIVU1
106 LQ
107 MADD
108 MADD1
109 MADDU
110 MADDU1
111 MFHI1
112 MFLO1
113 MFSA
114 MTHI1
115 MTLO1
116 MTSA
117 MTSAB
118 MTSAH
119 MULT
120 MULT1
121 MULTU
122 MULTU1
123 PABSH
124 PABSW
125 PADDB
126 PADDH
127 PADDSB
128 PADDSH
129 PADDSW
130 PADDUB
131 PADDUH
132 PADDUW
133 PADDW
134 PADSBH
135 PAND
136 PCEQB
137 PCEQH
138 PCEQW
139 PCGTB
140 PCGTH
141 PCGTW
142 PCPYH

143 PCPYLD
144 PCPYUD
145 PDIVBW
146 PDIVUW
147 PDIVW
148 PEXCH
149 PEXCW
150 PEXEH
151 PEXEW
152 PEXT5
153 PEXTLB
154 PEXTLH
155 PEXTLW
156 PEXTUB
157 PEXTUH
158 PEXTUW
159 PHMADH
160 PHMSBH
161 PINTEH
162 PINTH
163 PLZCW
164 PMADDH
165 PMADDUW
166 PMADDW
167 PMAXH
168 PMAXW
169 PMFHI
170 PMFHL.LH
171 PMFHL.LW
172 PMFHL.SH
173 PMFHL.SLW
174 PMFHL.UW
175 PMFLO
176 PMINH
177 PMINW
178 PMSUBH
179 PMSUBW
180 PMTHI
181 PMTHL.LW

182 PMTLO
183 PMULTH
184 PMULTUW
185 PMULTW
186 PNOR
187 POR
188 PPAC5
189 PPACB
190 PPACH
191 PPACW
192 PREVH
193 PROT3W
194 PSLLH
195 PSLLVW
196 PSLLW
197 PSRAH
198 PSRAVW
199 PSRAW
200 PSRLH
201 PSRLVW
202 PSRLW
203 PSUBB
204 PSUBH
205 PSUBSB
206 PSUBSH
207 PSUBSW
208 PSUBUB
209 PSUBUH
210 PSUBUW
211 PSUBW
212 PXOR
213 QFSRV
214 SQ

215 BC0F
216 BC0FL
217 BC0T
218 BC0TL
219 CACHE BFH
220 CACHE BHINBT
221 CACHE BXLBT
222 CACHE BXSBT
223 CACHE DHIN
224 CACHE DHWBIN
225 CACHE DHWOIN
226 CACHE DXIN
227 CACHE DXLDT
228 CACHE DXLTG
229 CACHE DXSDT
230 CACHE DXSTG
231 CACHE DXWBIN
232 CACHE IFL
233 CACHE IHIN
234 CACHE IXIN
235 CACHE IXLDT
236 CACHE IXLTG
237 CACHE IXSDT
238 CACHE IXSTG
239 DI
240 EI
241 ERET
242 MFBPC
243 MFC0
244 MFDAB
245 MFDABM
246 MFDVB
247 MFDVBM
248 MFIAB
249 MFIABM
250 MFPC
251 MFPS
252 MTBPC
253 MTC0

254 MTDAB
255 MTDABM
256 MTDVB
257 MTDVBM
258 MTIAB
259 MTIABM
260 MTPC
261 MTPS
262 TLBP
263 TLBR
264 TLBWI
265 TLBWR

266 ABS.S
267 ADD.S
268 ADD
269 ADDA.S
270 BC1F
271 BC1FL
272 FP False
273 BC1T
274 BC1TL
275 FP True Likely
276 CFC1
277 CTC1
278 CVT.S.W
279 CVT.W.S
280 DIV.S
281 LWC1
282 MADD.S
283 MADDA.S
284 MAX.S
285 MFC1
286 MIN.S
287 MOV.S
288 MSUB.S
289 MSUBA.S
290 MTC1
291 MUL.S
292 MULA.S
293 NEG.S
294 RSQRT.S
295 SQRT.S
296 SUB.S
297 SUBA.S
298 SWC1

299 BC2F
300 BC2FL
301 BC2T
302 BC2TL
303 CFC2
304 CTC2
305 LQC2
306 QMFC2
307 QMTC2
308 SQC2
309 VABS
310 VADD
311 VADDi
312 VADDq
313 VADDbc
314 VADDA
315 VADDAi
316 VADDAq
317 VADDAbc
318 VCALLMS
319 VCALLMSR
320 VCLIP
321 VDIV
322 VFTOI0
323 VFTOI4
324 VFTOI12
325 VFTOI15
326 VIADD
327 VIADDI
328 VIAND
329 VILWR
330 VIOR
331 VISUB
332 VISWR
333 VITOF0
334 VITOF4
335 VITOF12
336 VITOF15
337 VLQD

338 VLQI
339 VMADD
340 VMADDi
341 VMADDq
342 VMADDA
343 VMADDAi
344 VMADDAq
345 VMADDAbc
346 VMAX
347 VMAXi
348 VMAXbc
349 VMFIR
350 VMINI
351 VMINIi
352 VMINIbc
353 VMOVE
354 VMR32
355 VMSUB
356 VMSUBi
357 VMSUBq
358 VMSUBbc
359 VMSUBA
360 VMSUBAi
361 VMSUBAq
362 VMSUBAbc
363 VMTIR
364 VMUL
365 VMULi
366 VMULq
367 VMULbc
368 VMULA
369 VMULAi
370 VMULAq
371 VMULAbc
372 VNOP
373 VOPMULA
374 VOPMSUB
375 VRGET
376 VRINIT

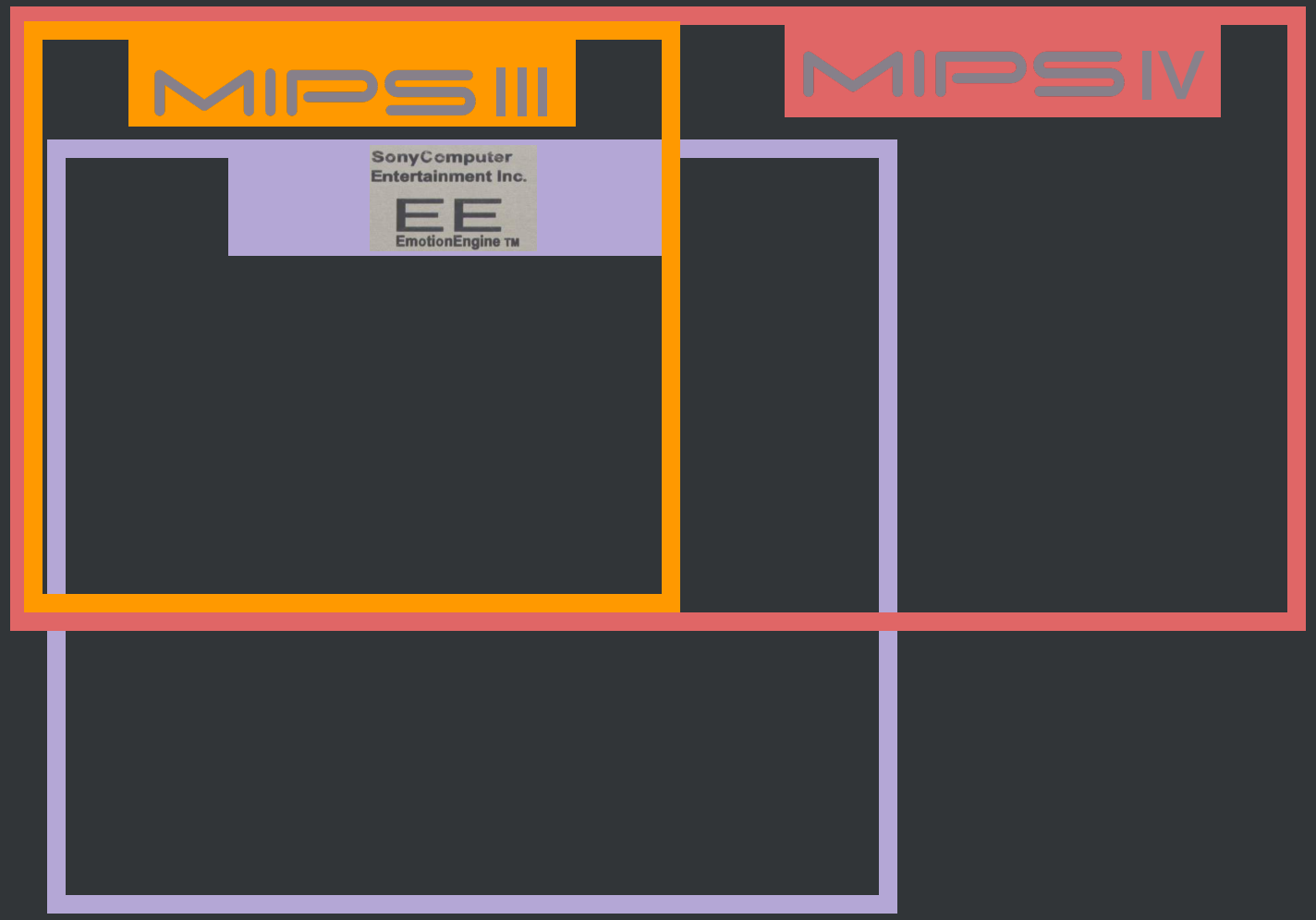
377 VRNEXT
378 VRSQRT
379 VRXOR
380 VSQD
381 VSQI
382 VSQRT
383 VSUB
384 VSUBi
385 VSUBq
386 VSUBbc
387 VSUBAi
388 VSUBAq
389 VSUBAbc
390 VWAITQ

EE - COP2

1 ADD
2 ADDI
3 ADDIU
4 ADDU
5 AND
6 ANDI
7 BEQ
8 BEQL
9 BGEZ
10 BGEZAL
11 BGEZALL
12 BGEZL
13 BGTZ
14 BGTZL
15 BLEZ
16 BLEZL
17 BLTZ
18 BLTZAL
19 BLTZALL
20 BLTZL
21 BNE
22 BNEL
23 BREAK
24 DADD
25 DADDI
26 DADDIU
27 DADDU
28 DIV
29 DIVU
30 DSLL
31 DSLL32
32 DSLLV
33 DSRA
34 DSRA32
35 DSRAV
36 DSRL
37 DSRL32
38 DSRLV
39 DSUB
40 DSUBU
41 JAL
42 JALR
43 JR
44 LB
45 LBU
46 LD
47 LDL
48 LDR
49 LH
50 LHU
51 LUI
52 LW
53 LWL
54 LWR
55 LWU
56 MFHI
57 MFLO
58 MOVN
59 MOVZ
60 MTHI
61 MTLO
62 MULT
63 MULTU
64 NOR
65 OR
66 ORI
67 PREF
68 SB
69 SD
70 SDL
71 SDR
72 SH
73 SLL
74 SLLV
75 SLT
76 SLTI
77 SLTIU
78 SLTU
79 SRA
80 SRAV
81 SRL
82 SRLV
83 SUB
84 SUBU
85 SW
86 SWL
87 SWR
88 SYNC.stype
89 SYSCALL
90 TEQ
91 TEQI
92 TGE
93 TGEI
94 TGEIU
95 TGEU
96 TLT
97 TLTI
98 TLTIU
99 TLTU
100 TNE
101 TNEI
102 XOR
103 XORI

This is the MIPS part!

104 DIV1
105 DIVU1
106 LQ
107 MADD
108 MADD1
109 MADDU
110 MADDU1
111 MFHI1
112 MFLO1
113 MFSA
114 MTHI1
115 MTLO1
116 MTSA
117 MTSAB
118 MTSAH
119 MULT
120 MULT1
121 MULTU
122 MULTU1
123 PABSH
124 PABSW
125 PADDB
126 PADDH
127 PADDSB
128 PADDSH
129 PADDSW
130 PADDUB
131 PADDUH
132 PADDUW
133 PADDW
134 PADSBH
135 PAND
136 PCEQB
137 PCEQH
138 PCEQW
139 PCGTB
140 PCGTH
141 PCGTW
142 PCPYH
143 PCPYLD
144 PCPYUD
145 PDIVBW
146 PDIVUW
147 PDIVW
148 PEXCH
149 PEXCW
150 PEXEH
151 PEXEW
152 PEXT5
153 PEXTLB
154 PEXTLH
155 PEXTLW
156 PEXTUB
157 PEXTUH
158 PEXTUW
159 PHMADH
160 PHMSBH
161 PINTEH
162 PINTH
163 PLZCW
164 PMADDH
165 PMADDUW
166 PMADDW
167 PMAXH
168 PMAXW
169 PMFHI
170 PMFHL.LH
171 PMFHL.LW
172 PMFHL.SH
173 PMFHL.SLW
174 PMFHL.UW
175 PMFLO
176 PMINH
177 PMINW
178 PMSUBH
179 PMSUBW
180 PMTHI
181 PMTHL.LW
182 PMTLO
183 PMULTH
184 PMULTUW
185 PMULTW
186 PNOR
187 POR
188 PPAC5
189 PPACB
190 PPACH
191 PPACW
192 PREVH
193 PROT3W
194 PSLLH
195 PSLLVW
196 PSLLW
197 PSRAH
198 PSRAVW
199 PSRAW
200 PSRLH
201 PSRLVW
202 PSRLW
203 PSUBB
204 PSUBH
205 PSUBSB
206 PSUBSH
207 PSUBSW
208 PSUBUB
209 PSUBUH
210 PSUBUW
211 PSUBW
212 PXOR
213 QFSRV
214 SQ
215 BC0F
216 BC0FL
217 BC0T
218 BC0TL
219 CACHE BFH
220 CACHE BHINBT
221 CACHE BXLBT
222 CACHE BXSBT
223 CACHE DHIN
224 CACHE DHWBIN
225 CACHE DHWOIN
226 CACHE DXIN
227 CACHE DXLDT
228 CACHE DXLTG
229 CACHE DXSDT
230 CACHE DXSTG
231 CACHE DXWBIN
232 CACHE IFL
233 CACHE IHIN
234 CACHE IXIN
235 CACHE IXLDT
236 CACHE IXLTG
237 CACHE IXSDT
238 CACHE IXSTG
239 DI
240 EI
241 ERET
242 MFBPC
243 MFC0
244 MFDAB
245 MFDABM
246 MFDVB
247 MFDVBM
248 MFIAB
249 MFIABM
250 MFPC
251 MFPS
252 MTBPC
253 MTC0
254 MTDAB
255 MTDABM
256 MTDVB
257 MTDVBM
258 MTIAB
259 MTIABM
260 MTPC
261 MTPS
262 TLBP
263 TLBR
264 TLBWI
265 TLBWR
266 ABS.S
267 ADD.S
268 ADD
269 ADDA.S
270 BC1F
271 BC1FL
272 FP False
273 BC1T
274 BC1TL
275 FP True Likely
276 CFC1
277 CTC1
278 CVT.S.W
279 CVT.W.S
280 DIV.S
281 LWC1
282 MADD.S
283 MADDA.S
284 MAX.S
285 MFC1
286 MIN.S
287 MOV.S
288 MSUB.S
289 MSUBA.S
290 MTC1
291 MUL.S
292 MULA.S
293 NEG.S
294 RSQRT.S
295 SQRT.S
296 SUB.S
297 SUBA.S
298 SWC1
299 BC2F
300 BC2FL
301 BC2T
302 BC2TL
303 CFC2
304 CTC2
305 LQC2
306 QMFC2
307 QMTC2
308 SQC2
309 VABS
310 VADD
311 VADDi
312 VADDq
313 VADDbc
314 VADDA
315 VADDAi
316 VADDAq
317 VADDAbc
318 VCALLMS
319 VCALLMSR
320 VCLIP
321 VDIV
322 VFTOI0
323 VFTOI4
324 VFTOI12
325 VFTOI15
326 VIADD
327 VIADDI
328 VIAND
329 VILWR
330 VIOR
331 VISUB
332 VISWR
333 VITOF0
334 VITOF4
335 VITOF12
336 VITOF15
337 VLQD
338 VLQI
339 VMADD
340 VMADDi
341 VMADDq
342 VMADDA
343 VMADDAi
344 VMADDAq
345 VMADDAbc
346 VMAX
347 VMAXi
348 VMAXbc
349 VMFIR
350 VMINI
351 VMINIi
352 VMINIbc
353 VMOVE
354 VMR32
355 VMSUB
356 VMSUBi
357 VMSUBq
358 VMSUBbc
359 VMSUBA
360 VMSUBAi
361 VMSUBAq
362 VMSUBAbc
363 VMTIR
364 VMUL
365 VMULi
366 VMULq
367 VMULbc
368 VMULA
369 VMULAi
370 VMULAq
371 VMULAbc
372 VNOP
373 VOPMULA
374 VOPMSUB
375 VRGET
376 VRINIT
377 VRNEXT
378 VRSQRT
379 VRXOR
380 VSQD
381 VSQI
382 VSQRT
383 VSUB
384 VSUBi
385 VSUBq
386 VSUBbc
387 VSUBAi
388 VSUBAq
389 VSUBAbc
390 VWAITQ


MIPS III
MIPS IV
Sony Computer
Entertainment Inc.
EE
EmotionEngine TM

Sony Computer Entertainment Inc.

EE

EmotionEngine ™

PIPELINE

DELAY SLOTS

```
1 lw t5,0x0(t7) ; t5 = MEM[t7]
2 jr t5 ; jump to t5
3 addiu t5,t5,4 ; t5+=4
```

PIPELINE

DELAY SLOTS

```
lw t5,0x0(t7) ; t5 = MEM[t7]
jr t5 ; jump to t5
addiu t5,t5,4 ; t5+=4
```

PIPELINE

DELAY SLOTS

```
lw t5,0x0(t7) ; t5 = MEM[t7]
jr t5 ; jump to t5
addiu t5,t5,4 ; t5+=4
```

PIPELINE

DELAY SLOTS

```
1 lw t5,0x0(t7) ; t5 = MEM[t7]
2 jr t5 ; jump to t5
3 addiu t5,t5,4 ; t5+=4
```

???

PIPELINE

PIPELINE

A CPU executes instructions by passing through multiple steps

PIPELINE

A CPU executes instructions by passing through multiple steps

We call those a pipeline

PIPELINE

PIPELINE

A CPU executes instructions by passing through multiple steps

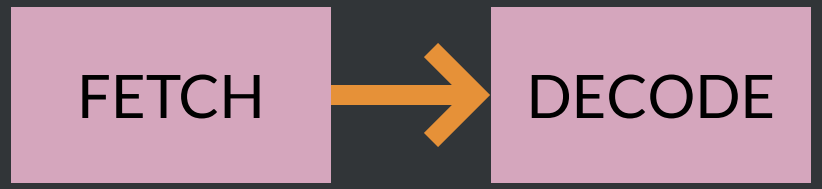
We call those a pipeline

FETCH

PIPELINE

A CPU executes instructions by passing through multiple steps

We call those a pipeline



PIPELINE

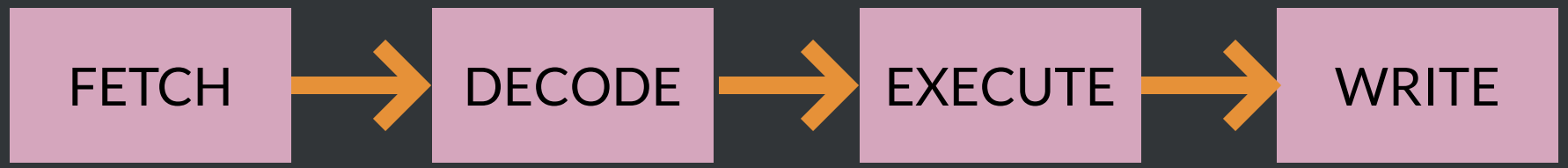
A CPU executes instructions by passing through multiple steps

We call those a pipeline

PIPELINE

A CPU executes instructions by passing through multiple steps

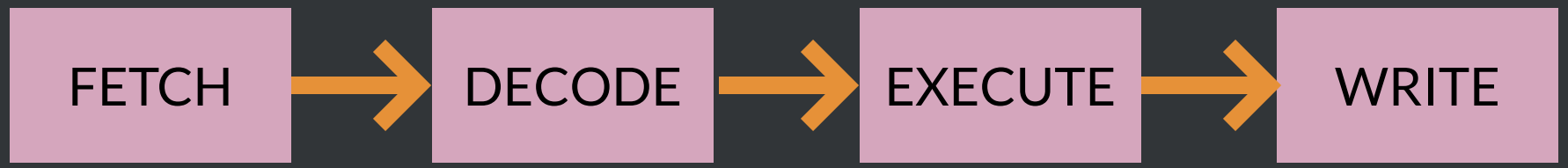
We call those a pipeline



PIPELINE

A CPU executes instructions by passing through multiple steps

We call those a pipeline

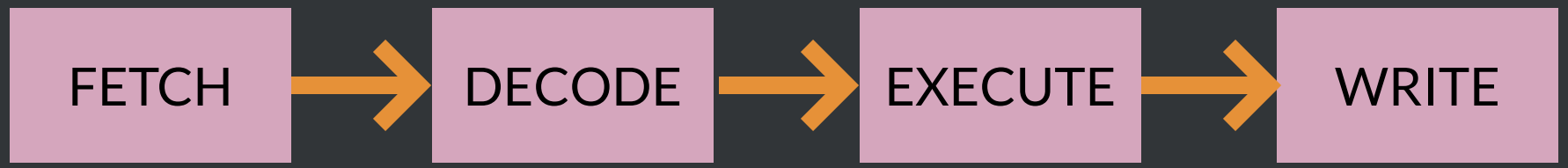


Not exactly true but it'll work for now

PIPELINE

A CPU executes instructions by passing through multiple steps

We call those a pipeline



Not exactly true but it'll work for now

The execute step will also be used as memory access

PIPELINE

PIPELINE

Here's an example

FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF

s3 = 00

FETCH → DECODE → EXECUTE → WRITE

78 00 b3 ff

PIPELINE

PIPELINE

Here's an example

FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF

s3 = 00

FETCH → DECODE → EXECUTE → WRITE

sd s3,0x78(sp)

PIPELINE

PIPELINE

Here's an example

FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF

s3 = 00

FETCH → DECODE → EXECUTE → WRITE

sd s3, FF

PIPELINE

PIPELINE

Here's an example

FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF
s3 = FF

FETCH → DECODE → EXECUTE → WRITE

sd s3,

PIPELINE

PIPELINE

Here's an example

FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF
s3 = FF

FETCH → DECODE → EXECUTE → WRITE

sd s3,

In reality all of those steps are executed in parallel on multiple instructions

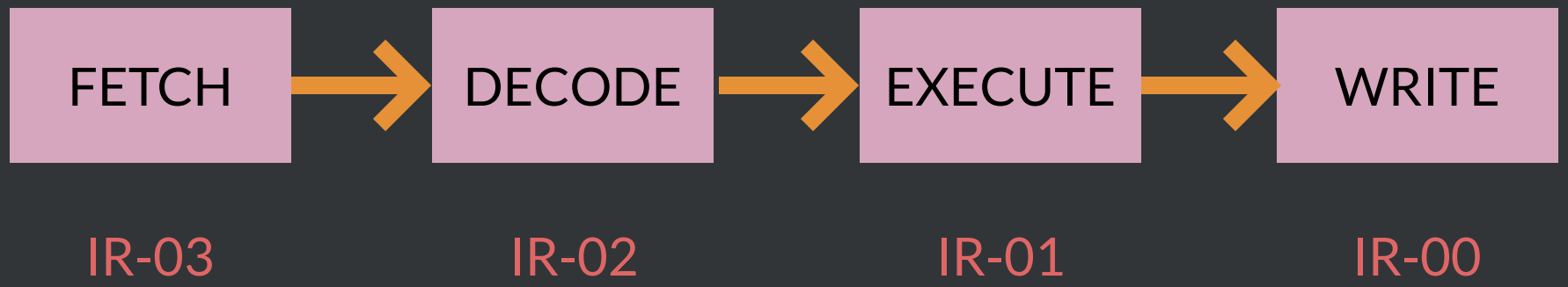
PIPELINE

PIPELINE

Here's an example

FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF

s3 = FF



IR = Instruction Register, current instruction

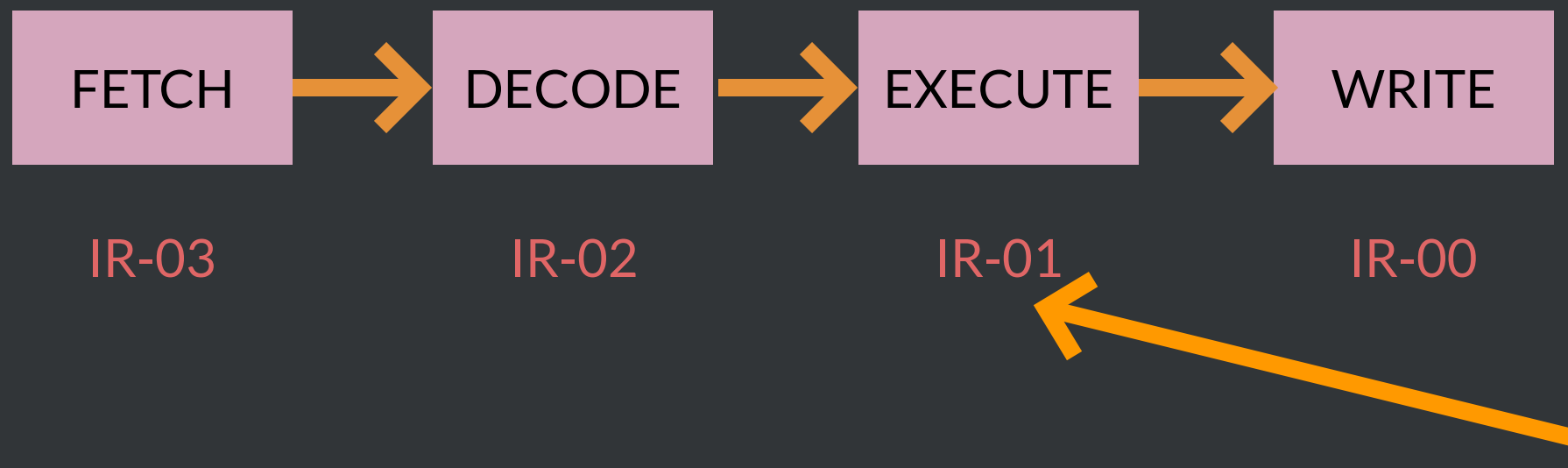
PIPELINE

PIPELINE

Here's an example

FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF

s3 = FF



What if this is a jump?

IR = Instruction Register, current instruction

PIPELINE

PIPELINE

Here's an example

FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF

s3 = FF

FETCH → DECODE → EXECUTE → WRITE

This happens after the jump!

jmp

IR-00

What if this is a jump?

IR = Instruction Register, current instruction

PIPELINE

PIPELINE

FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF
s3 = FF

Here's an example

Instead of wasting 2 steps, MIPS decided to execute an instruction out of order to waste 1

FETCH → DECODE → EXECUTE → WRITE

This happens after the jump!

jmp

IR-00

What if this is a jump?

IR = Instruction Register, current instruction

PIPELINE

PIPELINE

FF	FF	FF	FF	FF	FF	FF	FF
FF	FF	FF	FF	FF	FF	FF	FF
FF	FF	FF	FF	FF	FF	FF	FF
FF	FF	FF	FF	FF	FF	FF	
FF	FF	FF	FF	FF	FF	FF	FF

s3 = FF

Here's an example

Instead of wasting 2 steps, MIPS decided to execute an instruction out of order to waste 1

FETCH → DECODE → EXECUTE → WRITE

IR-02 jmp IR-00

What if this is a jump?

IR = Instruction Register, current instruction

PIPELINE

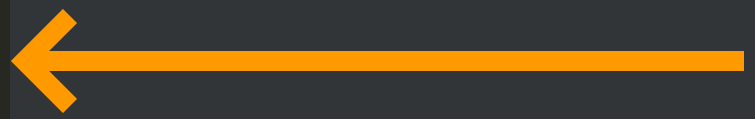
THE DELAY SLOTS STRIKES BACK

```
lui $a0, 0
li $v1, FlushCache
syscall
li $v1, ResetEE
```

PIPELINE

THE DELAY SLOTS STRIKES BACK

```
1 lui $a0, 0
2 li $v1, FlushCache
3 syscall
4 li $v1, ResetEE
```



PIPELINE

THE DELAY SLOTS STRIKES BACK

```
1 lui $a0, 0
2 li $v1, FlushCache
3 syscall
4 li $v1, ResetEE
```

PIPELINE

THE DELAY SLOTS STRIKES BACK

```
1 lui $a0, 0
2 li $v1, FlushCache
3 syscall
4 li $v1, ResetEE
```

PIPELINE

THE DELAY SLOTS STRIKES BACK

```
lui $a0, 0
li $v1, FlushCache
syscall
li $v1, ResetEE
```

¿¿¿

INTERRUPTS

PIPELINE

INTERRUPTS

syscall is like an interrupt instruction

PIPELINE

INTERRUPTS

syscall is like an interrupt instruction

The CPU switches to kernel mode and drops the entire pipeline

PIPELINE

INTERRUPTS

syscall is like an interrupt instruction

The CPU switches to kernel mode and drops the entire pipeline

Everything gets fetched back again after the syscall is done

COP

COPO

COP

COPO

Handles multiple system things:

COP

COPO

Handles multiple system things:

- Memory Management

COP

COPO

Handles multiple system things:

- Memory Management
- Exceptions

COP

COPO

Handles multiple system things:

- Memory Management
- Exceptions
- Debugging

COP

COPO

Handles multiple system things:

- Memory Management
- Exceptions
- Debugging
- Cache

COP

COP0

Handles multiple system things:

- Memory Management
- Exceptions
- Debugging
- Cache
- Interrupts! (Nice transition)

COP

COP1

COP

COP1

A Floating Point Unit (FPU)

COP

COP1

A Floating Point Unit (FPU)
(this thing)

0.39212301373481750488281 25
0x3ec8c459

COP

COP1

A Floating Point Unit (FPU)
(this thing)

0.39212301373481750488281 25
0x3ec8c459

Not IEEE 754 compliant!!

COP

COP1

A Floating Point Unit (FPU)
(this thing)

0.39212301373481750488281 25
0x3ec8c459

Not IEEE 754 compliant!!

Relevant list of features not implemented:

COP

COP1

A Floating Point Unit (FPU)
(this thing)

0.39212301373481750488281 25
0x3ec8c459

Not IEEE 754 compliant!!

Relevant list of features not implemented:

- NaN

COP

COP1

A Floating Point Unit (FPU)
(this thing)

0.39212301373481750488281 25
0x3ec8c459

Not IEEE 754 compliant!!

Relevant list of features not implemented:

- NaN
- Nearest roundings

COP

COP1

A Floating Point Unit (FPU)
(this thing)

0.39212301373481750488281 25
0x3ec8c459

Not IEEE 754 compliant!!

Relevant list of features not implemented:

- NaN
- Nearest roundings
- +/- ∞

COP

COP1

A Floating Point Unit (FPU)
(this thing)

0.39212301373481750488281 25
0x3ec8c459

Not IEEE 754 compliant!!

Relevant list of features not implemented:

- NaN
- Nearest roundings
- +/- ∞
- Exceptions

COP

COP1

A Floating Point Unit (FPU)
(this thing)

0.39212301373481750488281 25
0x3ec8c459

Not IEEE 754 compliant!!

Relevant list of features not implemented:

- NaN
- Nearest roundings
- +/- ∞
- Exceptions
- Denormalized numbers

COP

COP1

A Floating Point Unit (FPU)
(this thing)

0.39212301373481750488281 25
0x3ec8c459

Not IEEE 754 compliant!!

Relevant list of features not implemented:

- NaN
- Nearest roundings
- +/- ∞
- Exceptions
- Denormalized numbers

Result: an absolute pain in the ass to emulate

VPU

VPU

VPU

Two of them, composed of two things:

- A Vector Unit (VU)

VPU

Two of them, composed of two things:

- A Vector Unit (VU)
- A Vector Interface (VIF)

VPU

Two of them, composed of two things:

- A Vector Unit (VU)
- A Vector Interface (VIF)

VPU0 can either work as a COP or as a microprocessor

VPU

Two of them, composed of two things:

- A Vector Unit (VU)
- A Vector Interface (VIF)

VPU0 can either work as a COP or as a microprocessor

If it runs in COP(macro) mode, it will act as a superset of instructions for the EE core

VPU

Two of them, composed of two things:

- A Vector Unit (VU)
- A Vector Interface (VIF)

VPU0 can either work as a COP or as a microprocessor

If it runs in COP(macro) mode, it will act as a superset of instructions for the EE core

Otherwise it will execute instructions in parallel fed in a microprogram by the EE

VPU

VPU

VPU

VPU1 can transfer directly to the GS memory by using 2 methods:

VPU

VPU1 can transfer directly to the GS memory by using 2 methods:

- XGKICK (Path 1)

VPU

VPU1 can transfer directly to the GS memory by using 2 methods:

- XGKICK (Path 1)
- VIF1 (Path 2)

VPU

VPU1 can transfer directly to the GS memory by using 2 methods:

- XGKICK (Path 1)
- VIF1 (Path 2)

The EE and the VU1 uses a third method to transfer data to the GPU, the GIF

VPU

VPU1 can transfer directly to the GS memory by using 2 methods:

- XGKICK (Path 1)
- VIF1 (Path 2)

The EE and the VU1 uses a third method to transfer data to the GPU, the GIF

NB: Path 1 and Path 2 also use the GIF but have higher priority, confusing yet?

VPU - EXAMPLE

VPU - EXAMPLE

VPU

The EE sends the model data to the VIF

VPU

VPU - EXAMPLE

The EE sends the model data to the VIF

```
.align 0
;test.dae_mp1_pkt1.obj
;Automatically generated by kh2vif
;kh2vif by GovanifY ~ 2017
stcycl 01, 01

unpack[r] V4_32, 0, *
.int 1, 0, 0, 0
.int 36, 4, 54, 56
.int 0, 0, 0, 0
.int 14, 40, 0, 5
.EndUnpack

stcycl 01, 01

unpack[r] V2_16, 4, *
.short 2048, 0
.short 1024, 1024
.short 1024, 0
.short 1024, 3071
.short 2048, 2048
.short 2048, 3071
.short 3071, 2048
```

VPU - EXAMPLE

VPU

The EE sends the model data to the VIF

```
.align 0
;test.dae_mp1_pkt1.obj
;Automatically generated by kh2vif
;kh2vif by GovanifY ~ 2017
stcycl 01, 01

unpack[r] V4_32, 0, *
.int 1, 0, 0, 0
.int 36, 4, 54, 56
.int 0, 0, 0, 0
.int 14, 40, 0, 5
.EndUnpack

stcycl 01, 01

unpack[r] V2_16, 4, *
.short 2048, 0
.short 1024, 1024
.short 1024, 0
.short 1024, 3071
.short 2048, 2048
.short 2048, 3071
.short 3071, 2048
```

VIF1

The VIF1 executes the unpack commands and writes the data to its memory

VPU

VPU - EXAMPLE

The EE sends the model data to the VIF

```
.align 0
;test.dae_mp1_pkt1.obj
;Automatically generated by kh2vif
;kh2vif by GovanifY ~ 2017
stcycl 01, 01

unpack[r] V4_32, 0, *
.int 1, 0, 0, 0
.int 36, 4, 54, 56
.int 0, 0, 0, 0
.int 14, 40, 0, 5
.EndUnpack

stcycl 01, 01

unpack[r] V2_16, 4, *
.short 2048, 0
.short 1024, 1024
.short 1024, 0
.short 1024, 3071
.short 2048, 2048
.short 2048, 3071
.short 3071, 2048
```

The VU1 transforms the data and calculate relative positions

VU1

VIF1

The VIF1 executes the unpack commands and writes the data to its memory

VPU

VPU - EXAMPLE

The EE sends the model data to the VIF

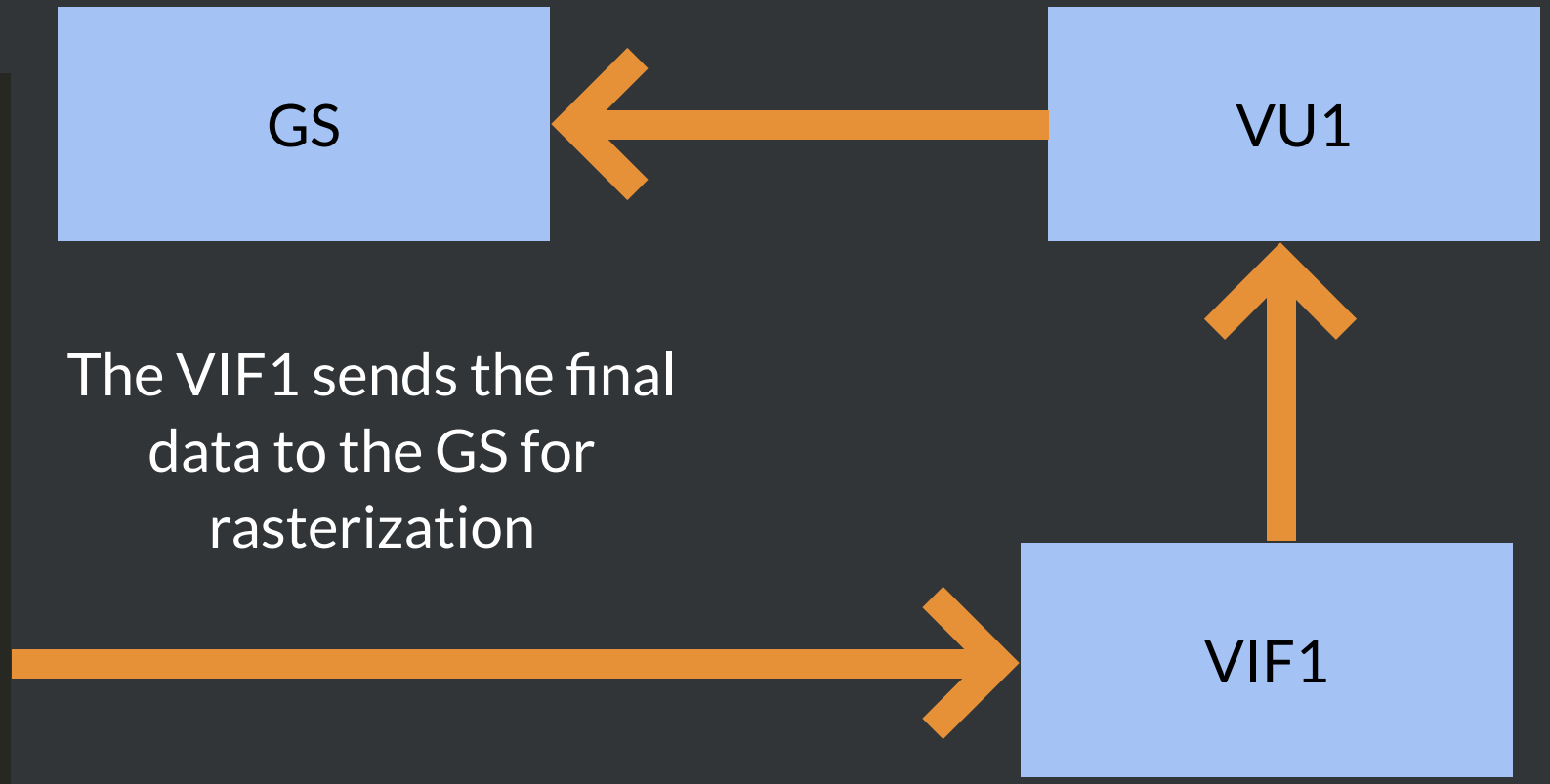
```
.align 0
;test.dae_mp1_pkt1.obj
;Automatically generated by kh2vif
;kh2vif by GovanifY ~ 2017
stcycl 01, 01

unpack[r] V4_32, 0, *
.int 1, 0, 0, 0
.int 36, 4, 54, 56
.int 0, 0, 0, 0
.int 14, 40, 0, 5
.EndUnpack

stcycl 01, 01

unpack[r] V2_16, 4, *
.short 2048, 0
.short 1024, 1024
.short 1024, 0
.short 1024, 3071
.short 2048, 2048
.short 2048, 3071
.short 3071, 2048
```

The VU1 transforms the data and calculate relative positions



The VIF1 sends the final data to the GS for rasterization

The VIF1 executes the unpack commands and writes the data to its memory

SPU

SPU2

SPU

SPU2

Based on the PS1 SPU, but with 2 cores!

SPU

SPU2

Based on the PS1 SPU, but with 2 cores!

SPU

SPU2

Based on the PS1 SPU, but with 2 cores!

Has customizable IRQ!!

SPU

SPU2

Based on the PS1 SPU, but with 2 cores!

Has customizable IRQ!!

Games use them as highly precise interrupts by setting an IRQ at a write-back address used during the mixing stage

SPU

SPU2

Based on the PS1 SPU, but with 2 cores!

Has customizable IRQ!!

Games use them as highly precise interrupts by setting an IRQ at a write-back address used during the mixing stage

The mixer has a sample rate of 48kHZ in PS2 mode, 44.1 in PS1 compatible mode

SPU

SPU2

SPU

SPU2

Also has a Schroeder Reverberator!

Uses 4 parallel comb filters in a rotating buffer

SPU

SPU2

Also has a Schroeder Reverberator!

Uses 4 parallel comb filters in a rotating buffer

Adds gain, then mixes back with the original input, rewriting the rotating buffer in the process

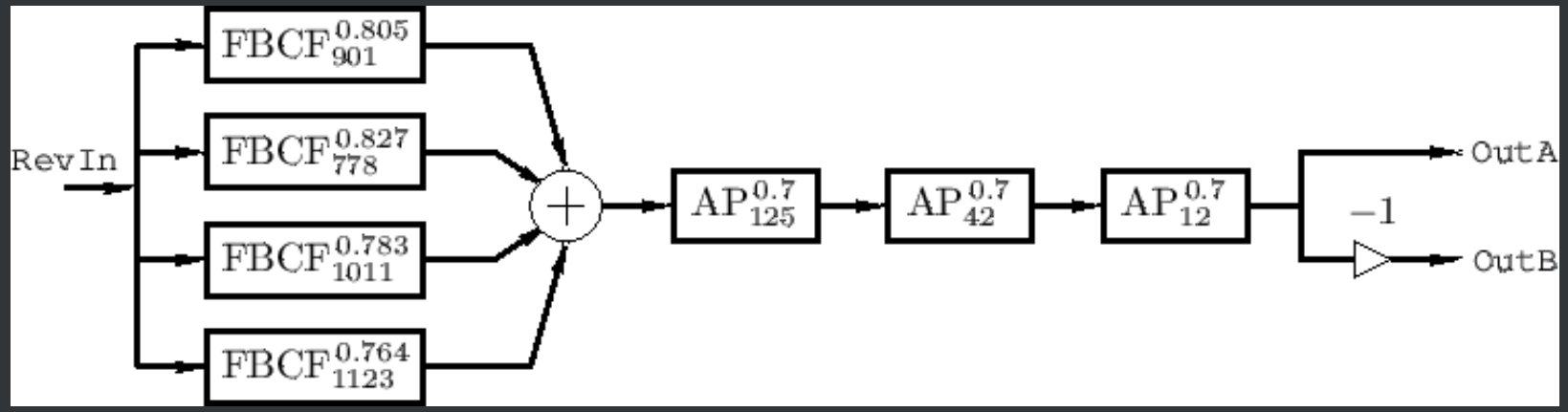
SPU

SPU2

Also has a Schroeder Reverberator!

Uses 4 parallel comb filters in a rotating buffer

Adds gain, then mixes back with the original input, rewriting the rotating buffer in the process



WHAT IS THE IOP

IOP

WHAT IS THE IOP

Good question! It's a MIPS-based processor...

WHAT IS THE IOP

Good question! It's a MIPS-based processor...
... or PowerPC.

WHAT IS THE IOP

Good question! It's a MIPS-based processor...
... or PowerPC.

wait...**WHAT**?

WHAT IS THE IOP

Good question! It's a MIPS-based processor...
... or PowerPC.

wait...**WHAT**?

... Let's ignore that for now.

IOP

WHAT IS THE IOP

... Let's ignore that for now.

WHAT IS THE IOP

Good question! It's a MIPS-based processor...
... Let's ignore that for now.

It is the PS1 CPU, just repurposed in order to handle all the I/O, devices and drivers in the PS2.

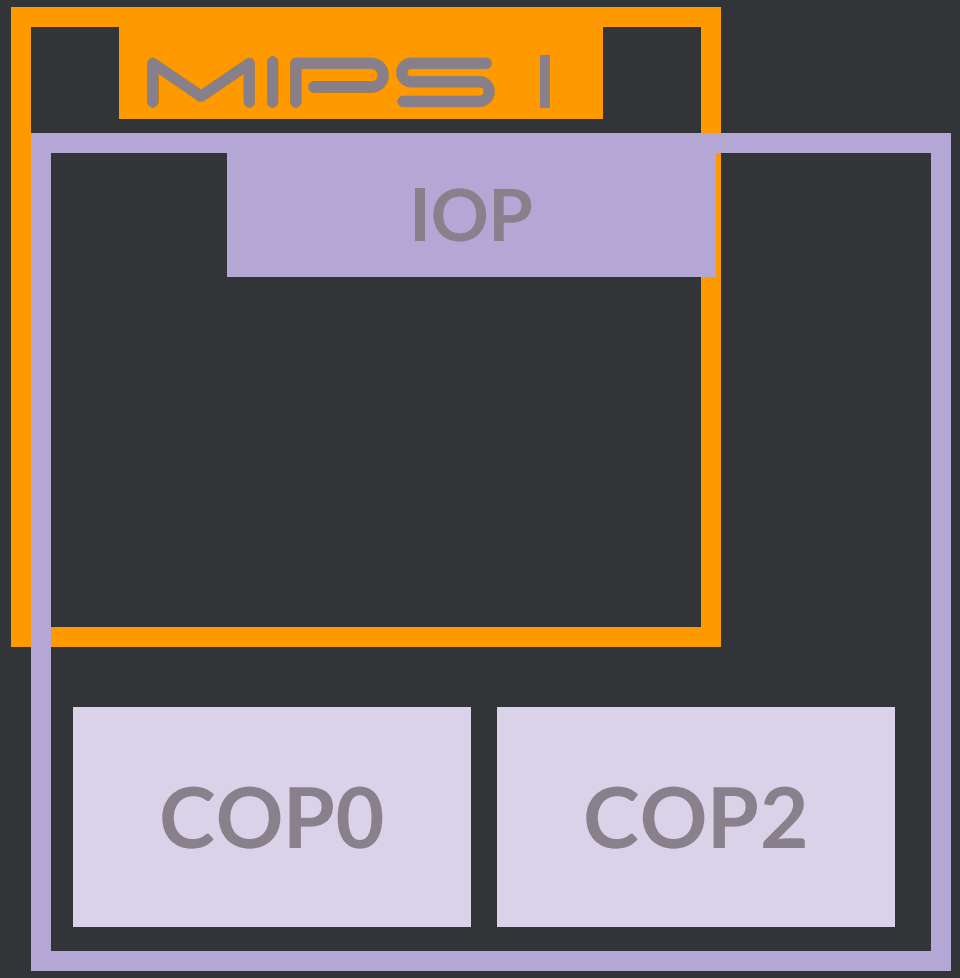
WHAT IS THE IOP

Good question! It's a MIPS-based processor...
... Let's ignore that for now.

It is the PS1 CPU, just repurposed in order to handle all the I/O, devices and drivers in the PS2.

The EE and the IOP communicate through the Subsystem Interface (SIF).

MIPS I


MIPS I
IOP
COP0
COP2

WHAT IS THE IOP

WHAT IS THE IOP

A MIPS I "compatible" CPUs with 2 COP

WHAT IS THE IOP

A MIPS I "compatible" CPUs with 2 COP

- COP0: System Management

WHAT IS THE IOP

A MIPS I "compatible" CPUs with 2 COP

- COP0: System Management
- COP2: Geometry Transformation Engine (GTE)

WHAT IS THE IOP

A MIPS I "compatible" CPUs with 2 COP

- COP0: System Management
- **COP1: ???**
- COP2: Geometry Transformation Engine (GTE)

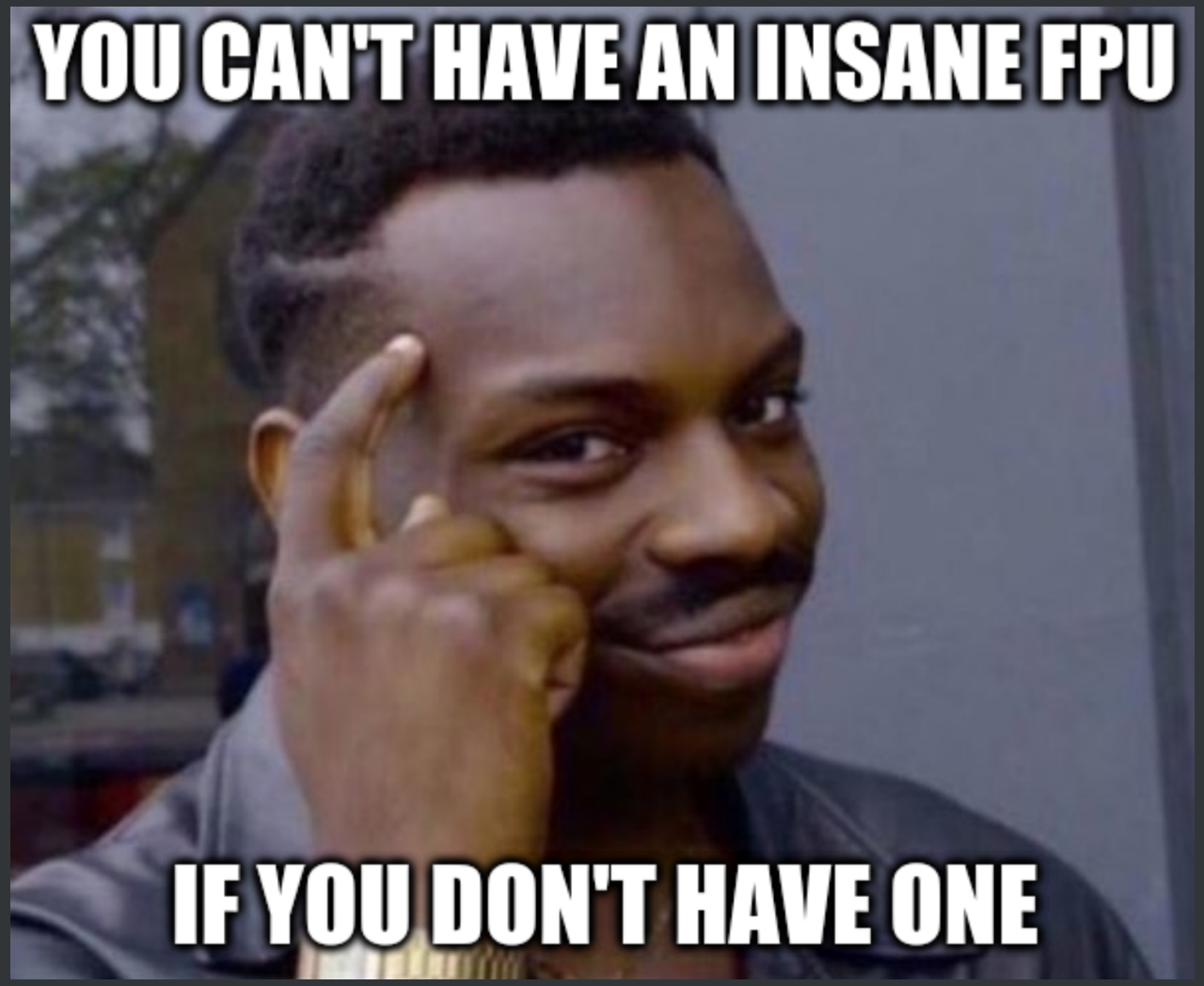
WHAT IS THE IOP

A MIPS I "compatible" CPUs with 2 COP

- COP0: System Management
- **COP1: ???**
- COP2: Geometry Transformation Engine (GTE)

Sony doesn't know how to count

IOP



IOP

ONE LAST THING

Dynamic Memory Allocation and the Heap

Memory allocation functions allow the programmer to increase the depth and variety of the game world whilst making the best use of the PlayStation's relatively modest memory size.

In C, memory allocation is achieved using the *malloc* and *free* functions. Unfortunately there have been several problems with these functions on PlayStation.

The standard *malloc/free* combination supplied as part of *libc* fragments memory due to a bug in the free function. This means that large chunks on the machine memory become inaccessible even though they are not holding any valid data.

IOP

ONE LAST THING

Dynamic Memory Allocation and the Heap

Memory allocation functions allow the programmer to increase the depth and variety of the game world whilst making the best use of the PlayStation's relatively modest memory size.

In C, memory allocation is achieved using the *malloc* and *free* functions. Unfortunately there have been several problems with these functions on PlayStation.

The standard *malloc/free* combination supplied as part of *libc* fragments memory due to a bug in the free function. This means that large chunks on the machine memory become inaccessible even though they are not holding any valid data.

PS1: Bring out your own f---ing chip and system libraries

DECKARD

DECKARD

Meet PS2 Slim hardware!

DECKARD

Slim, right?

Meet PS2 Slim hardware!

DECKARD

Slim, right?

Wanna know why?

Meet PS2 Slim hardware!

DECKARD


SonyComputer
Entertainment Inc.
CXD2976GB
©&Ⓜ2007SCEI
733L86N
754531

DECKARD

The combined EE+GS+RDRAM+DRAM found in the SCPH-7900x and SCPH-9000x series

DECKARD

A PS2 ON-A-CHIP IS NOT ENOUGH

DECKARD

A PS2 ON-A-CHIP IS NOT ENOUGH

Meet deckard!

DECKARD

A PS2 ON-A-CHIP IS NOT ENOUGH

Meet deckard!

A PowerPC based replacement for the IOP

DECKARD

A PS2 ON-A-CHIP IS NOT ENOUGH

Meet deckard!

A PowerPC based replacement for the IOP

Emulates PS1 features through software

DECKARD

A PS2 ON-A-CHIP IS NOT ENOUGH

Meet deckard!

A PowerPC based replacement for the IOP

Emulates PS1 features through software

Fortunately we don't care about it, we are writing an emulator, not trying to emulate the emulator emulating the console :D

DRM

COPY PROTECTION

DRM

COPY PROTECTION

About time we talk about it

DRM

COPY PROTECTION

About time we talk about it

Essentially a mod of PS1's copy protection

DRM

COPY PROTECTION

About time we talk about it

Essentially a mod of PS1's copy protection

The PS1 replaced CD's ATIP (which is a sinusoidal constant of ~22kHZ) by their own region specific constant

DRM

COPY PROTECTION

About time we talk about it

Essentially a mod of PS1's copy protection

The PS1 replaced CD's ATIP (which is a sinusoidal constant of ~22kHZ) by their own region specific constant

The ATIP is normally used by players to synchronize their laser's timing

DRM

COPY PROTECTION

About time we talk about it

Essentially a mod of PS1's copy protection

The PS1 replaced CD's ATIP (which is a sinusoidal constant of ~22kHZ) by their own region specific constant

The ATIP is normally used by players to synchronize their laser's timing

Data can also be stored by modulating the ATIP +/- 1kHZ!

DRM

COPY PROTECTION

The ATIP is around there

DRM

COPY PROTECTION

DRM

COPY PROTECTION

The PS2 instead stores the Title ID of the disc in it

DRM

COPY PROTECTION

The PS2 instead stores the Title ID of the disc in it

Mechacon then derives an encryption key out of the Title ID which will be used to decrypt and verify the disc

DRM

COPY PROTECTION

The PS2 instead stores the Title ID of the disc in it

Mechacon then derives an encryption key out of the Title ID which will be used to decrypt and verify the disc

It will then proceed to decrypt the "PlayStation 2" logo you see at each boot once sent to it

DRM

COPY PROTECTION

The PS2 instead stores the Title ID of the disc in it

Mechacon then derives an encryption key out of the Title ID which will be used to decrypt and verify the disc

It will then proceed to decrypt the "PlayStation 2" logo you see at each boot once sent to it

...But we can completely ignore this by skipping the verification logic in the BIOS!

DRM

COPY PROTECTION

DRM

COPY PROTECTION

Sony tries to make this harder by making it harder to power on the mechacon

DRM

COPY PROTECTION

Sony tries to make this harder by making it harder to power on the mechacon

...forgetting you could just dump the BIOS out of your flash chip and reverse engineer it

DRM

COPY PROTECTION

Sony tries to make this harder by making it harder to power on the mechacon

...forgetting you could just dump the BIOS out of your flash chip and reverse engineer it

.....and that the bootloader verifies the integrity of your BIOS with a simple CRC which is prone to collisions

DRM

COPY PROTECTION

Sony tries to make this harder by making it harder to power on the mechacon

...forgetting you could just dump the BIOS out of your flash chip and reverse engineer it

.....and that the bootloader verifies the integrity of your BIOS with a simple CRC which is prone to collisions

The mechacon is essentially a security processor that you can completely ignore and useless in functionality

```
1  void mechaconAuth()
2  {
3    int k;
4
5    while (cdvdRead(0x17) != 0x40) {;}
6
7
8    cdvdWrite(0x17, 0);
9
10   cdvdWrite(0x16, 0x80);
11   while (cdvdRead(0x16) != 0x80) {;}
12   while (cdvdRead(0x17) != 0x40)
13   {
14     cdvdRead(0x18);
15   }
16
17   while (cdvdRead(0x17) != 0x40) {;}
18   cdvdWrite(0x16, 0x81);
19   while (cdvdRead(0x16) != 0x81) {;}
20   while (cdvdRead(0x17) != 0x40)
21   {
22     cdvdRead(0x18);
23   }
24
25
26   while (cdvdRead(0x17) != 0x40) {;}
27   for (k = 0; k < 16; k++)
28   {
29     cdvdWrite(0x17, 0xff);
30   }
```

```c
void mechaconAuth()
{
  int k;

  while (cdvdRead(0x17) != 0x40) {;}


  cdvdWrite(0x17, 0);

  cdvdWrite(0x16, 0x80);
  while (cdvdRead(0x16) != 0x80) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }

  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x81);
  while (cdvdRead(0x16) != 0x81) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 16; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x82);
  while (cdvdRead(0x16) != 0x82) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 8; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x83);
  while (cdvdRead(0x16) != 0x83) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x8f);
  while (cdvdRead(0x16) != 0x8f) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x84);
  while (cdvdRead(0x16) != 0x84) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x85);
  while (cdvdRead(0x16) != 0x85) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }

  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 16; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x86);
  while (cdvdRead(0x16) != 0x86) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 8; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x87);
  while (cdvdRead(0x16) != 0x87) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x88);
  while (cdvdRead(0x16) != 0x88) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x80);
  while (cdvdRead(0x16) != 0x80) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x17, 0x08);

  cdvdWrite(0x16, 0x80);
  while (cdvdRead(0x16) != 0x80) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x17, 0x08);

  cdvdWrite(0x16, 0x81);
  while (cdvdRead(0x16) != 0x81) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 16; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x82);
  while (cdvdRead(0x16) != 0x82) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 8; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x83);
  while (cdvdRead(0x16) != 0x83) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x8f);
  while (cdvdRead(0x16) != 0x8f) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x84);
  while (cdvdRead(0x16) != 0x84) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x85);
  while (cdvdRead(0x16) != 0x85) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }

  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 16; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x86);
  while (cdvdRead(0x16) != 0x86) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 8; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x87);
  while (cdvdRead(0x16) != 0x87) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x8f);
  while (cdvdRead(0x16) != 0x8f) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x88);
  while (cdvdRead(0x16) != 0x88) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }
}
```

```c
void mechaconAuth()
{
  int k;

  while (cdvdRead(0x17) != 0x40) {;}

  cdvdWrite(0x17, 0);

  cdvdWrite(0x16, 0x80);
  while (cdvdRead(0x16) != 0x80) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }

  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x81);
  while (cdvdRead(0x16) != 0x81) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 16; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x82);
  while (cdvdRead(0x16) != 0x82) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 8; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x83);
  while (cdvdRead(0x16) != 0x83) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x8f);
  while (cdvdRead(0x16) != 0x8f) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x84);
  while (cdvdRead(0x16) != 0x84) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x85);
  while (cdvdRead(0x16) != 0x85) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }

  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 16; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x86);
  while (cdvdRead(0x16) != 0x86) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 8; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x87);
  while (cdvdRead(0x16) != 0x87) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }
```

```c

  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x8f);
  while (cdvdRead(0x16) != 0x8f) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x88);
  while (cdvdRead(0x16) != 0x88) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x17, 0x08);

  cdvdWrite(0x16, 0x80);
  while (cdvdRead(0x16) != 0x80) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x17, 0x08);

  cdvdWrite(0x16, 0x81);
  while (cdvdRead(0x16) != 0x81) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 16; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x82);
  while (cdvdRead(0x16) != 0x82) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 8; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x83);
  while (cdvdRead(0x16) != 0x83) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x8f);
  while (cdvdRead(0x16) != 0x8f) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x84);
  while (cdvdRead(0x16) != 0x84) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x85);
  while (cdvdRead(0x16) != 0x85) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 16; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x86);
  while (cdvdRead(0x16) != 0x86) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 8; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x87);
  while (cdvdRead(0x16) != 0x87) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x8f);
  while (cdvdRead(0x16) != 0x8f) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x88);
  while (cdvdRead(0x16) != 0x88) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }
}
```

Good thing we can just ignore it when emulating!

```
void mechaconAuth()
{
  int k;

  while (cdvdRead(0x17) != 0x40) {;}

  cdvdWrite(0x17, 0);

  cdvdWrite(0x16, 0x80);
  while (cdvdRead(0x16) != 0x80) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }

  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x81);
  while (cdvdRead(0x16) != 0x81) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 16; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x82);
  while (cdvdRead(0x16) != 0x82) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 8; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x83);
  while (cdvdRead(0x16) != 0x83) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x8f);
  while (cdvdRead(0x16) != 0x8f) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x84);
  while (cdvdRead(0x16) != 0x84) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x85);
  while (cdvdRead(0x16) != 0x85) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }

  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 16; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x86);
  while (cdvdRead(0x16) != 0x86) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 8; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x87);
  while (cdvdRead(0x16) != 0x87) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }
```

```

  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x8f);
  while (cdvdRead(0x16) != 0x8f) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x88);
  while (cdvdRead(0x16) != 0x88) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x17, 0x08);

  cdvdWrite(0x16, 0x80);
  while (cdvdRead(0x16) != 0x80) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x17, 0x08);

  cdvdWrite(0x16, 0x81);
  while (cdvdRead(0x16) != 0x81) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 16; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x82);
  while (cdvdRead(0x16) != 0x82) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 8; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x83);
  while (cdvdRead(0x16) != 0x83) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x8f);
  while (cdvdRead(0x16) != 0x8f) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x84);
  while (cdvdRead(0x16) != 0x84) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x85);
  while (cdvdRead(0x16) != 0x85) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }

  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 16; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x86);
  while (cdvdRead(0x16) != 0x86) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 8; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x87);
  while (cdvdRead(0x16) != 0x87) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x8f);
  while (cdvdRead(0x16) != 0x8f) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x88);
  while (cdvdRead(0x16) != 0x88) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }
}
```

Good thing we can just ignore it when emulating!

```
case 0x80:                    // secrman: __mechacon_auth_0x80
        SetResultSize(1);      //in:1
        cdvd.mg_datatype = 0; //data
        cdvd.Result[0] = 0;
        break;

case 0x81:                    // secrman: __mechacon_auth_0x81
        SetResultSize(1);      //in:1
        cdvd.mg_datatype = 0; //data
        cdvd.Result[0] = 0;
        break;

case 0x82:                 // secrman: __mechacon_auth_0x82
        SetResultSize(1); //in:16
        cdvd.Result[0] = 0;
        break;
```

```
void mechaconAuth()
{
  int k;

  while (cdvdRead(0x17) != 0x40) {;}

  cdvdWrite(0x17, 0);

  cdvdWrite(0x16, 0x80);
  while (cdvdRead(0x16) != 0x80) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }

  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x81);
  while (cdvdRead(0x16) != 0x81) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 16; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x82);
  while (cdvdRead(0x16) != 0x82) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 8; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x83);
  while (cdvdRead(0x16) != 0x83) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x8f);
  while (cdvdRead(0x16) != 0x8f) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x84);
  while (cdvdRead(0x16) != 0x84) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x85);
  while (cdvdRead(0x16) != 0x85) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }

  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 16; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x86);
  while (cdvdRead(0x16) != 0x86) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 8; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x87);
  while (cdvdRead(0x16) != 0x87) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }
```

```

  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x8f);
  while (cdvdRead(0x16) != 0x8f) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x88);
  while (cdvdRead(0x16) != 0x88) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x17, 0x08);

  cdvdWrite(0x16, 0x80);
  while (cdvdRead(0x16) != 0x80) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x17, 0x08);

  cdvdWrite(0x16, 0x81);
  while (cdvdRead(0x16) != 0x81) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 16; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x82);
  while (cdvdRead(0x16) != 0x82) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 8; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x83);
  while (cdvdRead(0x16) != 0x83) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x8f);
  while (cdvdRead(0x16) != 0x8f) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x84);
  while (cdvdRead(0x16) != 0x84) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x85);
  while (cdvdRead(0x16) != 0x85) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }

  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 16; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x86);
  while (cdvdRead(0x16) != 0x86) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  for (k = 0; k < 8; k++)
  {
    cdvdWrite(0x17, 0xff);
  }

  cdvdWrite(0x16, 0x87);
  while (cdvdRead(0x16) != 0x87) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x8f);
  while (cdvdRead(0x16) != 0x8f) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }


  while (cdvdRead(0x17) != 0x40) {;}
  cdvdWrite(0x16, 0x88);
  while (cdvdRead(0x16) != 0x88) {;}
  while (cdvdRead(0x17) != 0x40)
  {
    cdvdRead(0x18);
  }
}
```

Good thing we can just ignore it when emulating!

```
case 0x80:                    // secrman: __mechacon_auth_0x80
        SetResultSize(1);      //in:1
        cdvd.mg_datatype = 0; //data
        cdvd.Result[0] = 0;
        break;

case 0x81:                    // secrman: __mechacon_auth_0x81
        SetResultSize(1);      //in:1
        cdvd.mg_datatype = 0; //data
        cdvd.Result[0] = 0;
        break;

case 0x82:                // secrman: __mechacon_auth_0x82
        SetResultSize(1); //in:16
        cdvd.Result[0] = 0;
        break;
```

Just have to make sure to return the nice values for the BIOS

DRM

COPY PROTECTION

DRM

COPY PROTECTION

What would be Sony's copy protection without trademark infringement?

DRM

COPY PROTECTION

What would be Sony's copy protection
without trademark infringement?

DRM

COPY PROTECTION

What would be Sony's copy protection
without trademark infringement?

DRM

COPY PROTECTION

What would be Sony's copy protection
without trademark infringement?


```
// letters = SLES, numbers = 54232
key[0] = ((numbers &  0x1F) <<  3) | ((0x0FFFFFFF & letters) >> 25);
key[1] = ( numbers          >> 10) | ((0x0FFFFFFF & letters) <<  7);
key[2] = ((numbers & 0x3E0) >>  2) | 0x04;
```

DRM

COPY PROTECTION

What would be Sony's copy protection without trademark infringement?


```
// letters = SLES, numbers = 54232
key[0] = ((numbers &  0x1F) <<  3) | ((0x0FFFFFFF & letters) >> 25);
key[1] = ( numbers          >> 10) | ((0x0FFFFFFF & letters) <<  7);
key[2] = ((numbers & 0x3E0) >>  2) | 0x04;
```

```
for(int i=0; i<12*2048; i++)
{
    logo[i] = ((logo[i]<<5)|(logo[i]>>3)) ^ magicNum;
}
```

DRM

COPY PROTECTION

What would be Sony's copy protection without trademark infringement?


```
// letters = SLES, numbers = 54232
key[0] = ((numbers &  0x1F) <<  3) | ((0x0FFFFFFF & letters) >> 25);
key[1] = ( numbers          >> 10) | ((0x0FFFFFFF & letters) <<  7);
key[2] = ((numbers & 0x3E0) >>  2) | 0x04;
```

```
for(int i=0; i<12*2048; i++)
{
    logo[i] = ((logo[i]<<5)|(logo[i]>>3)) ^ magicNum;
}
```

Also differs between regions

DRM

COPY PROTECTION

11101010

DRM

COPY PROTECTION

```
for(int i=0; i<12*2048; i++)
{
    logo[i] = ((logo[i]<<5)|(logo[i]>>3)) ^ magicNum;
}
```

11101010

DRM

COPY PROTECTION

```
for(int i=0; i<12*2048; i++)
{
    logo[i] = ((logo[i]<<5)|(logo[i]>>3)) ^ magicNum;
}
```

01011101

DRM

COPY PROTECTION

```
for(int i=0; i<12*2048; i++)
{
    logo[i] = ((logo[i]<<5)|(logo[i]>>3)) ^ magicNum;
}
```

01011101

^

11110000

DRM

COPY PROTECTION

```
for(int i=0; i<12*2048; i++)
{
    logo[i] = ((logo[i]<<5)|(logo[i]>>3)) ^ magicNum;
}
```

01011101

^

11110000

=

00011010

DRM

COPY PROTECTION

DRM

COPY PROTECTION

The key can be either calculated from the Title ID

DRM

COPY PROTECTION

The key can be either calculated from the Title ID

...or guessed by reading any 00 encrypted byte

DRM

COPY PROTECTION

The key can be either calculated from the Title ID

...or guessed by reading any 00 encrypted byte

00 ^ XX = XX

DRM

COPY PROTECTION

The key can be either calculated from the Title ID

...or guessed by reading any 00 encrypted byte

00 ^ XX = XX

The first byte of the logo is always 00

DRM

COPY PROTECTION

The key can be either calculated from the Title ID

...or guessed by reading any 00 encrypted byte

00 ^ XX = XX

The first byte of the logo is always 00

The 12 first sectors are dedicated to this, The next 2 are for Master Drives, and the last 2 are unused

DRM

COPY PROTECTION

DRM

COPY PROTECTION

Unhappy of having encrypted content which you can decrypt by simply reading its first byte Sony added a more convoluted protection mechanism called MagicGate to secure its memory cards

DRM

COPY PROTECTION

Unhappy of having encrypted content which you can decrypt by simply reading its first byte Sony added a more convoluted protection mechanism called MagicGate to secure its memory cards

You can obviously ask nicely the mechacon to sign and access memory cards for you, but that's not fun

DRM

MAGICGATE

DRM

MAGICGATE

MagicGate uses DES

DRM

MAGICGATE

MagicGate uses DES

Best public cryptanalysis

DES has been considered insecure right from the start because of the feasilibity of brute-force attacks[1] Such attacks have been demonstrated in practice (see EFF DES cracker)

DRM

MAGICGATE

MagicGate uses DES

Best public cryptanalysis

DES has been considered insecure right from the start because of the feasilibity of brute-force attacks[1] Such attacks have been demonstrated in practice (see EFF DES cracker)

Oh boy

DRM

MAGICGATE

MagicGate uses 3DES

Best public cryptanalysis

DES has been considered insecure right from the start because of the feasilibity of brute-force attacks[1] Such attacks have been demonstrated in practice (see EFF DES cracker)

Oh boy

DRM

MAGICGATE

MagicGate uses 3DES

Best public cryptanalysis

Lucks: 2^{32} known plaintexts, 2^{113} operations including 2^{90} DES encryptions, 2^{88} memory; Biham: find one of 2^{28} target keys with a handful of chosen plaintexts per key and 2^{84} encryptions

Oh boy

DRM

MAGICGATE

MagicGate uses 3DES

Best public cryptanalysis

Lucks: 2^{32} known plaintexts, 2^{113} operations including 2^{90} DES encryptions, 2^{88} memory; Biham: find one of 2^{28} target keys with a handful of chosen plaintexts per key and 2^{84} encryptions

...but with only 2 keys of security

DRM

MAGICGATE

DRM

MAGICGATE

Their 3DES implementation changes the key schedule slightly

DRM

MAGICGATE

Their 3DES implementation changes the key schedule slightly

They use it in CBC mode as a challenge reply nonce based cryptosystem

DRM

MAGICGATE

Their 3DES implementation changes the key schedule slightly

They use it in CBC mode as a challenge reply nonce based cryptosystem

1. We ask the memory card for some IV and its identifier

DRM

MAGICGATE

Their 3DES implementation changes the key schedule slightly

They use it in CBC mode as a challenge reply nonce based cryptosystem

1. We ask the memory card for some IV and its identifier
2. We derive a unique key based on this

DRM

MAGICGATE

Their 3DES implementation changes the key schedule slightly

They use it in CBC mode as a challenge reply nonce based cryptosystem

1. We ask the memory card for some IV and its identifier
2. We derive a unique key based on this
3. We ask the memory card for a nonce it generated

DRM

MAGICGATE

Their 3DES implementation changes the key schedule slightly

They use it in CBC mode as a challenge reply nonce based cryptosystem

1. We ask the memory card for some IV and its identifier
2. We derive a unique key based on this
3. We ask the memory card for a nonce it generated
4. We generate our nonce

DRM

MAGICGATE

Their 3DES implementation changes the key schedule slightly

They use it in CBC mode as a challenge reply nonce based cryptosystem

1. We ask the memory card for some IV and its identifier
2. We derive a unique key based on this
3. We ask the memory card for a nonce it generated
4. We generate our nonce
5. We generate a challenge: OurNonce|CardNonce|CardIV encrypted with the Unique Key we calculated and using a built-in IV

DRM

MAGICGATE

Their 3DES implementation changes the key schedule slightly

They use it in CBC mode as a challenge reply nonce based cryptosystem

1. We ask the memory card for some IV and its identifier
2. We derive a unique key based on this
3. We ask the memory card for a nonce it generated
4. We generate our nonce
5. We generate a challenge: OurNonce|CardNonce|CardIV encrypted with the Unique Key we calculated and using a built-in IV
6. The memory card decrypts our challenge and rebuilds another: CardNonce|MechaNonce|SessionKey, using the IV of step 5

DRM

MAGICGATE

Their 3DES implementation changes the key schedule slightly

They use it in CBC mode as a challenge reply nonce based cryptosystem

1. We ask the memory card for some IV and its identifier
2. We derive a unique key based on this
3. We ask the memory card for a nonce it generated
4. We generate our nonce
5. We generate a challenge: OurNonce|CardNonce|CardIV encrypted with the Unique Key we calculated and using a built-in IV
6. The memory card decrypts our challenge and rebuilds another: CardNonce|MechaNonce|SessionKey, using the IV of step 5
7. The SessionKey will now be used as a Key Encryption Key

DRM

MAGICGATE

DRM

MAGICGATE

This implementation has multiple issues:

- We can pull off chosen plaintext attacks by MITMing the mechacon and the memory card

DRM

MAGICGATE

This implementation has multiple issues:

- We can pull off chosen plaintext attacks by MITMing the mechacon and the memory card
- The IV used for the challenge is baked in and will never change.

DRM

MAGICGATE

This implementation has multiple issues:

- We can pull off chosen plaintext attacks by MITMing the mechacon and the memory card
- The IV used for the challenge is baked in and will never change.
- We can force Mechacon to keep reusing the same "unique" encryption key by resending the same CardIV and CardID

DRM

MAGICGATE

This implementation has multiple issues:

- We can pull off chosen plaintext attacks by MITMing the mechacon and the memory card
- The IV used for the challenge is baked in and will never change.
- We can force Mechacon to keep reusing the same "unique" encryption key by resending the same CardIV and CardID
- We can arbitrarily replace CardID and CardIV in any communication while keeping the same unique key(!)

DRM

MAGICGATE

This implementation has multiple issues:

- We can pull off chosen plaintext attacks by MITMing the mechacon and the memory card
- The IV used for the challenge is baked in and will never change.
- We can force Mechacon to keep reusing the same "unique" encryption key by resending the same CardIV and CardID
- We can arbitrarily replace CardID and CardIV in any communication while keeping the same unique key(!)
- ...not sensible to replay attacks, unlikely to be an oracle

DRM

MAGICGATE

This implementation has multiple issues:

- We can pull off chosen plaintext attacks by MITMing the mechacon and the memory card
- The IV used for the challenge is baked in and will never change.
- We can force Mechacon to keep reusing the same "unique" encryption key by resending the same CardIV and CardID
- We can arbitrarily replace CardID and CardIV in any communication while keeping the same unique key(!)
- ...not sensible to replay attacks, unlikely to be an oracle

Let's be smarter!

DRM

MAGICGATE

DRM

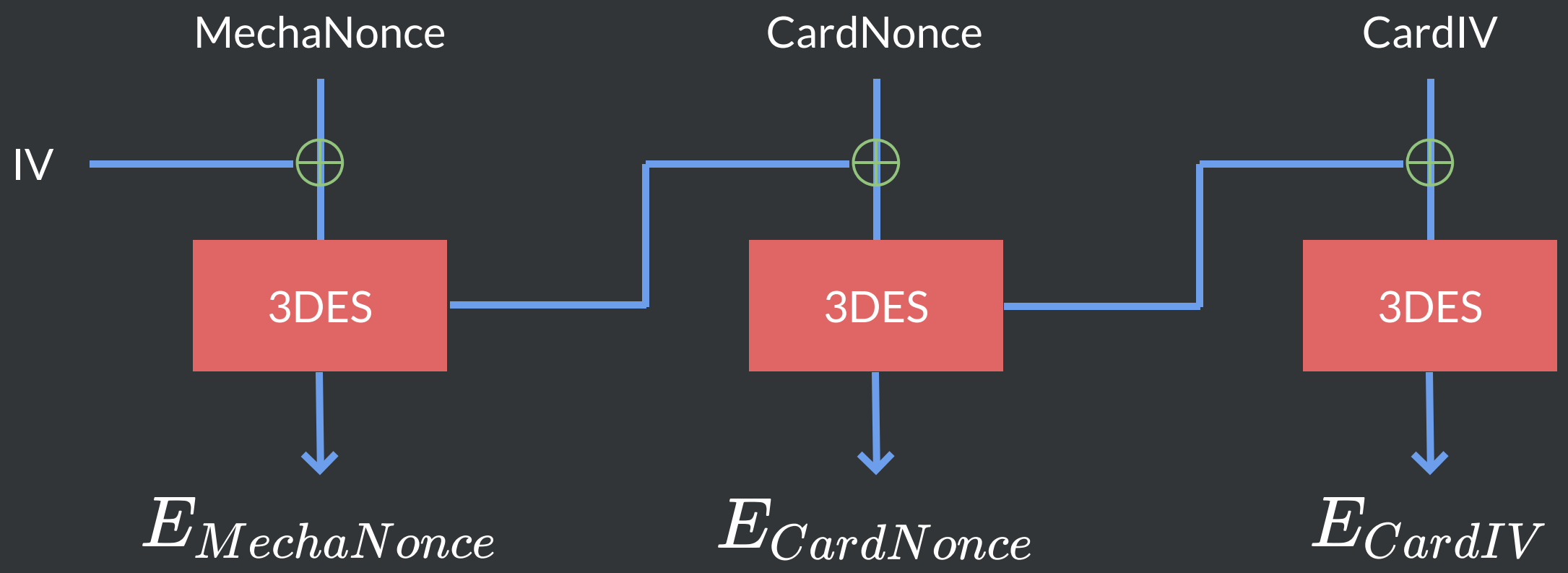
MAGICGATE

Mechacon challenge:

DRM

MAGICGATE

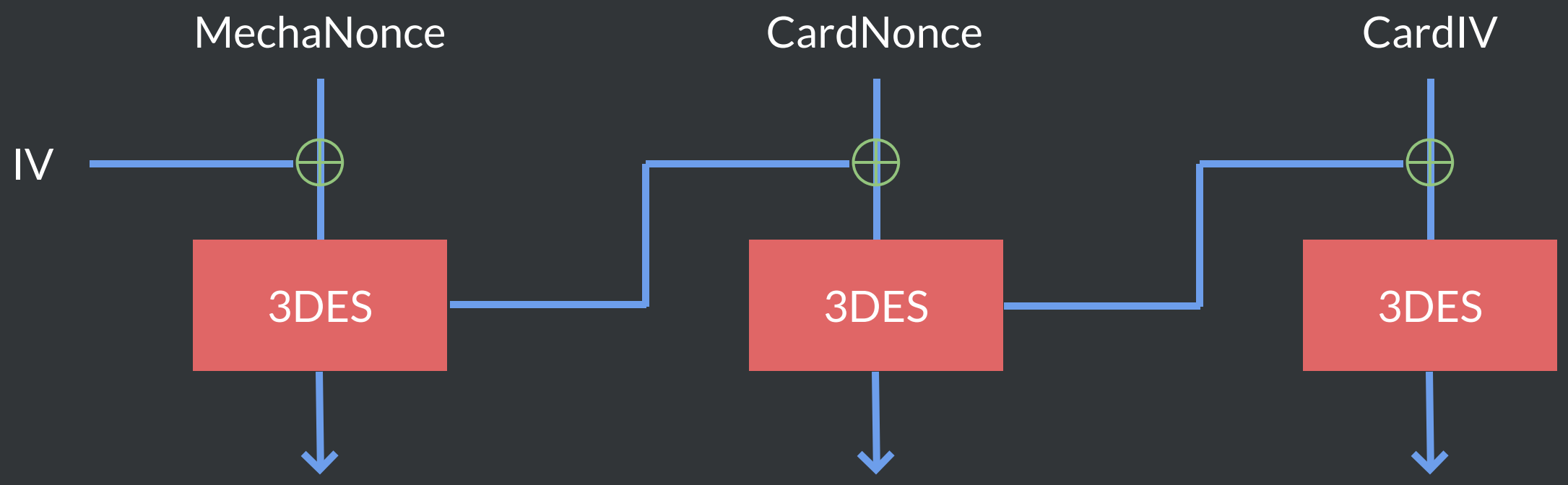
Mechacon challenge:



DRM

MAGICGATE

Mechacon challenge:



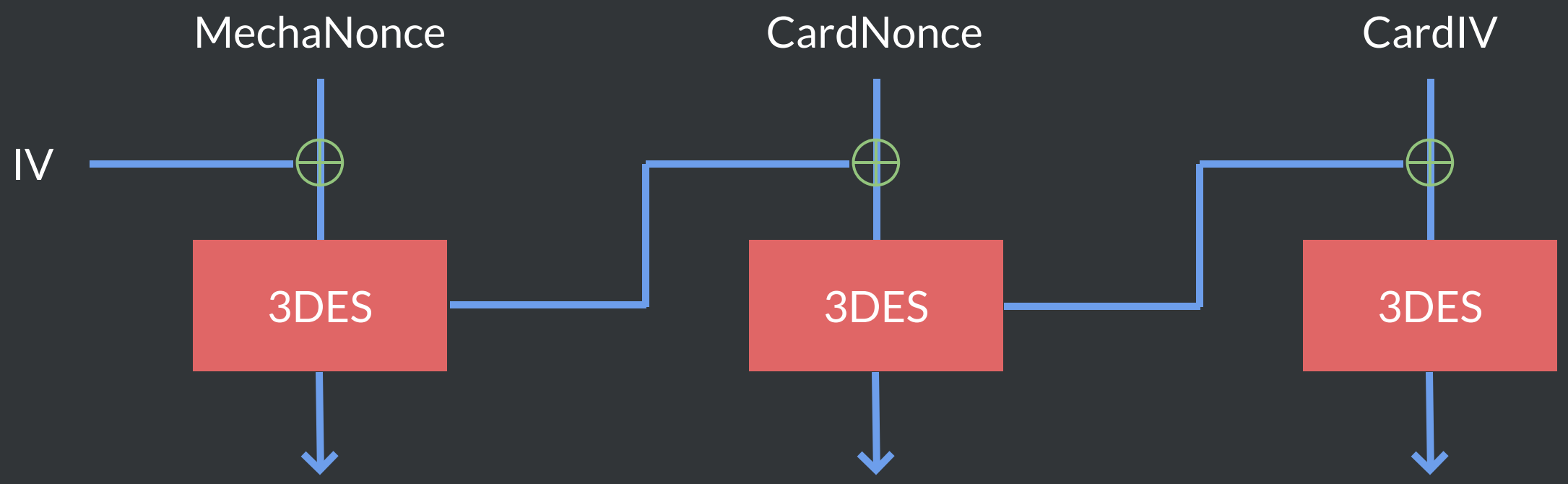
$$E_{MechaNonce} \qquad E_{CardNonce} \qquad E_{CardIV}$$

$$E_{CardIv} = E(CardIv \oplus E_{CardNonce}, K)$$

DRM

MAGICGATE

Mechacon challenge:



$$E_{MechaNonce} \qquad E_{CardNonce} \qquad E_{CardIV}$$

$$E_{CardIv} = E(CardIv \oplus E_{CardNonce}, K)$$

We can always predict everything but K so we can generate infinitely many known plaintext!

DRM

MAGICGATE

DRM

MAGICGATE

We can thus pull off a Linear Cryptanalysis attack on DES with our known plaintext dictionary

DRM

MAGICGATE

We can thus pull off a Linear Cryptanalysis attack on DES with our known plaintext dictionary

Matsui's attack can break it using 2^47 plaintext and was published in 1993. MagicGate was published in 1999.

DRM

MAGICGATE

We can thus pull off a Linear Cryptanalysis attack on DES with our known plaintext dictionary

Matsui's attack can break it using 2^47 plaintext and was published in 1993. MagicGate was published in 1999.

Biryukov et al's attack only requires 2^41 and was released in 2004.

DRM

MAGICGATE

We can thus pull off a Linear Cryptanalysis attack on DES with our known plaintext dictionary

Matsui's attack can break it using 2^47 plaintext and was published in 1993. MagicGate was published in 1999.

Biryukov et al's attack only requires 2^41 and was released in 2004.

On Multiple Linear Approximations

https://doi.org/10.1007/978-3-540-28628-8_1
hint: sci-hub

DRM

MAGICGATE

We can thus pull off a Linear Cryptanalysis attack on DES with our known plaintext dictionary

Matsui's attack can break it using 2^47 plaintext and was published in 1993. MagicGate was published in 1999.

Biryukov et al's attack only requires 2^41 and was released in 2004.

On Multiple Linear Approximations

https://doi.org/10.1007/978-3-540-28628-8_1
hint: sci-hub

But this only applies to DES!

DRM

MAGICGATE

DRM

MAGICGATE

Sony uses 3DES with a 2 key scheme, using the two keys on three encryption steps in this order:

DRM

MAGICGATE

Sony uses 3DES with a 2 key scheme, using the two keys on three encryption steps in this order:

$$k_1, k_2, k_1$$

DRM

MAGICGATE

Sony uses 3DES with a 2 key scheme, using the two keys on three encryption steps in this order:

$$k_1, k_2, k_1$$

An incorrect order could make a meet-in-the-middle attack possible, but unfortunately for us no can do here

DRM

MAGICGATE

Sony uses 3DES with a 2 key scheme, using the two keys on three encryption steps in this order:

$$k_1, k_2, k_1$$

An incorrect order could make a meet-in-the-middle attack possible, but unfortunately for us no can do here

Van Oorschot's attack based on Merkle is a known plaintext attack on 3DES with two triples which is now probably achievable by a dedicated adversary

DRM

MAGICGATE

Sony uses 3DES with a 2 key scheme, using the two keys on three encryption steps in this order:

$$k_1, k_2, k_1$$

An incorrect order could make a meet-in-the-middle attack possible, but unfortunately for us no can do here

Van Oorschot's attack based on Merkle is a known plaintext attack on 3DES with two triples which is now probably achievable by a dedicated adversary

A known-plaintext attack on two-key triple encryption

https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.66.6575

DRM

MAGICGATE

DRM

MAGICGATE

There are a few other details like Content Keys being derived after that, or the Memory Card replacing the Session Key by its own Storage Key once stored, but they are all vulnerable to this same attack.

DRM

MAGICGATE

There are a few other details like Content Keys being derived after that, or the Memory Card replacing the Session Key by its own Storage Key once stored, but they are all vulnerable to this same attack.

In the end we can extract all keys from mechacon blindly without using nitric acid!
Although I am unsure which is costlier nowadays

DRM

MAGICGATE

There are a few other details like Content Keys being derived after that, or the Memory Card replacing the Session Key by its own Storage Key once stored, but they are all vulnerable to this same attack.

In the end we can extract all keys from mechacon blindly without using nitric acid!
Although I am unsure which is costlier nowadays

...Or we can reverse engineer Sony's PS2 emulator which also includes the entire MagicGate algorithm to work with memory card adaptors

DRM

MAGICGATE

There are a few other details like Content Keys being derived after that, or the Memory Card replacing the Session Key by its own Storage Key once stored, but they are all vulnerable to this same attack.

In the end we can extract all keys from mechacon blindly without using nitric acid!
Although I am unsure which is costlier nowadays

...Or we can reverse engineer Sony's PS2 emulator which also includes the entire MagicGate algorithm to work with memory card adaptors

Oops^2

DRM

MAGICGATE

DRM

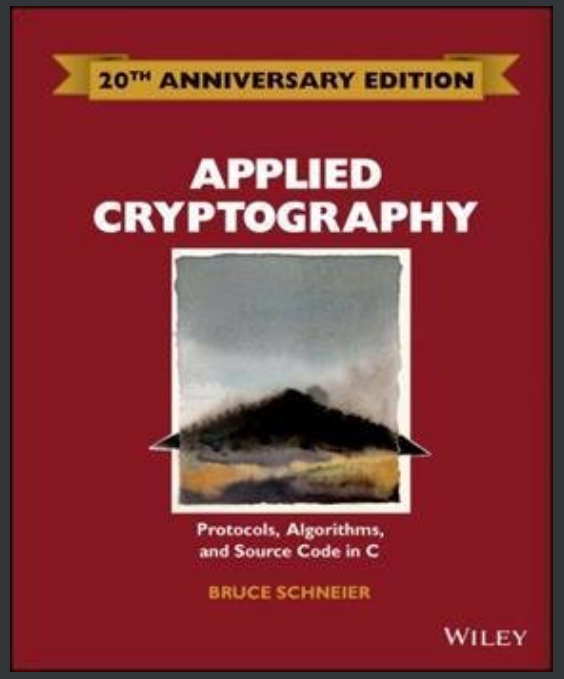
MAGICGATE

~~Sony,~~ here are some book recommendations if you want to study cryptography/DES a bit more

DRM

MAGICGATE

~~Sony,~~ here are some book recommendations if you want to study cryptography/DES a bit more

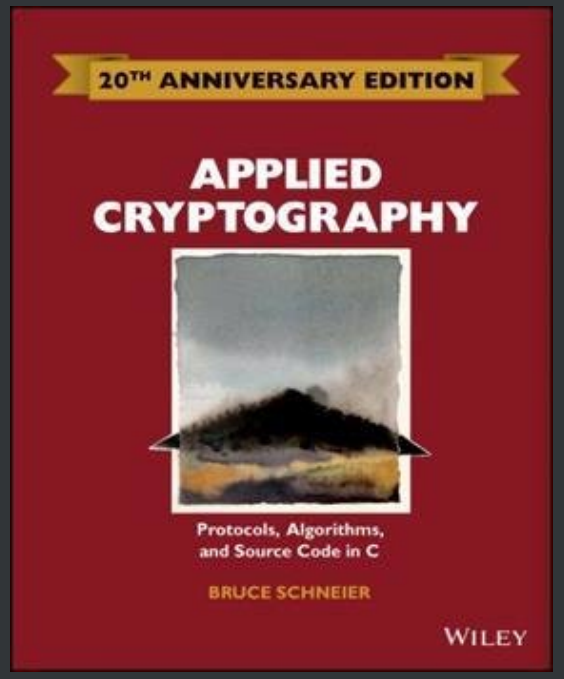


Applied Cryptography, Bruce Schneier

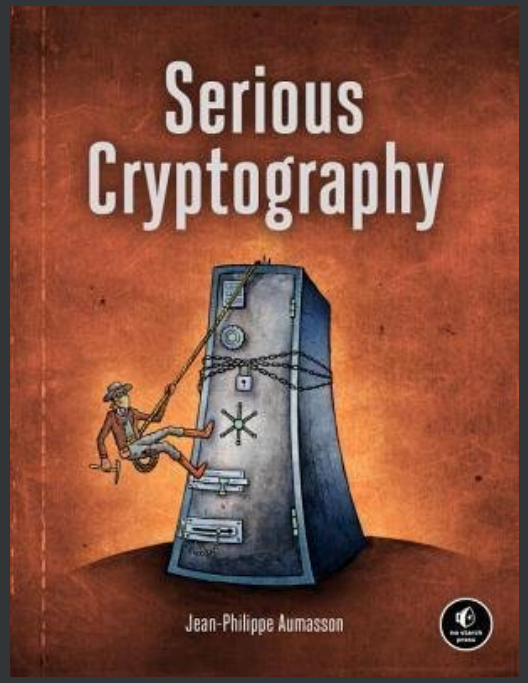
DRM

MAGICGATE

~~Sony,~~ here are some book recommendations if you want to study cryptography/DES a bit more



Applied Cryptography, Bruce Schneier

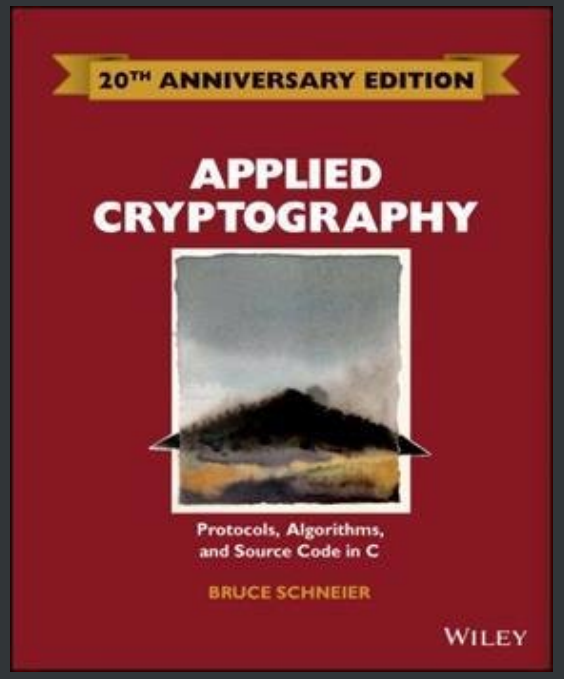


Serious Cryptography
Jean-Philippe Aumasson

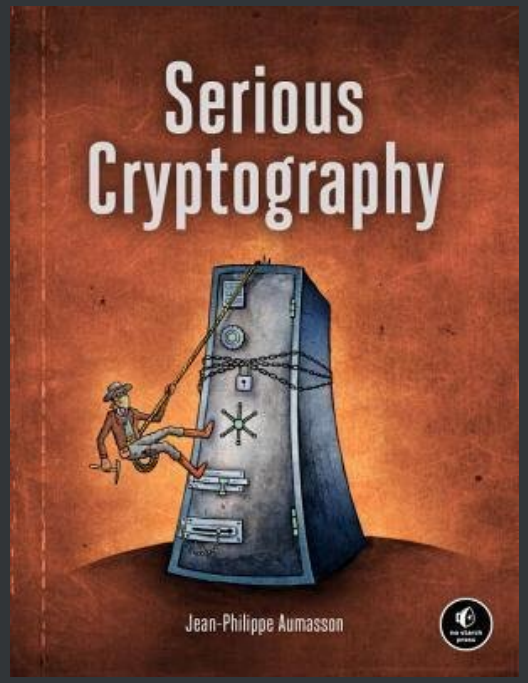
DRM

MAGICGATE

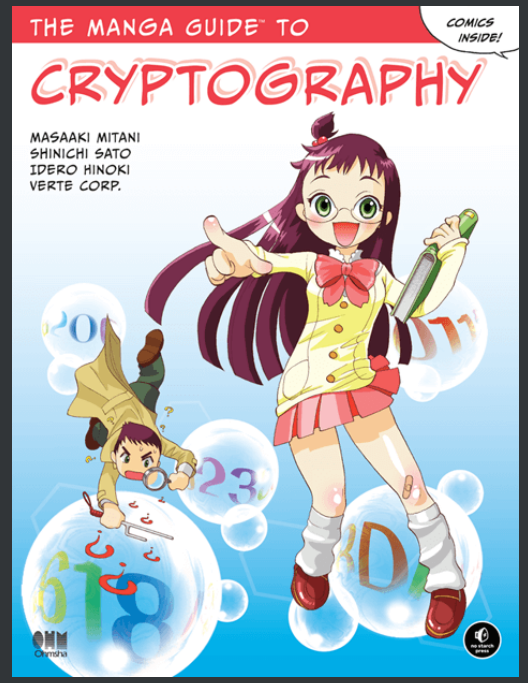
~~Sony,~~ here are some book recommendations if you want to study cryptography/DES a bit more



Applied Cryptography, Bruce Schneier



Serious Cryptography
Jean-Philippe Aumasson



The Manga Guide to Cryptography,
Masaaki, Shinichi, Idero, Verte et al.

GS

WHAT IS THE GS

GS

WHAT IS THE GS

A rasterizer

GS

WHAT IS THE GS

A rasterizer

That's it!!!

GS

WHAT IS THE GS

A rasterizer

That's it!!!

X	Y	Z	W
-1.0	-1.0	0.0	1.0
1.0	-1.0	0.0	1.0
0.0	1.0	0.0	1.0

GS

WHAT IS THE GS

A rasterizer

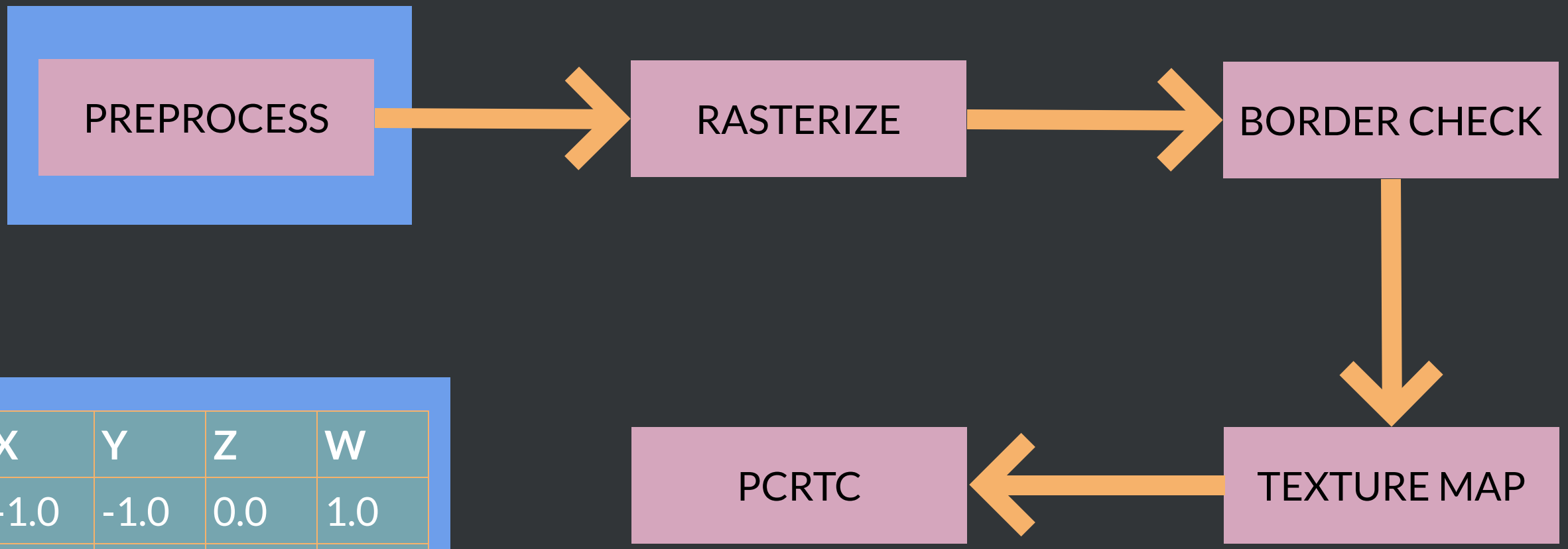
That's it!!!

X	Y	Z	W
-1.0	-1.0	0.0	1.0
1.0	-1.0	0.0	1.0
0.0	1.0	0.0	1.0

Draws Internally into a Framebuffer. A part of the GS called PCRTC then outputs it to your TV

GS

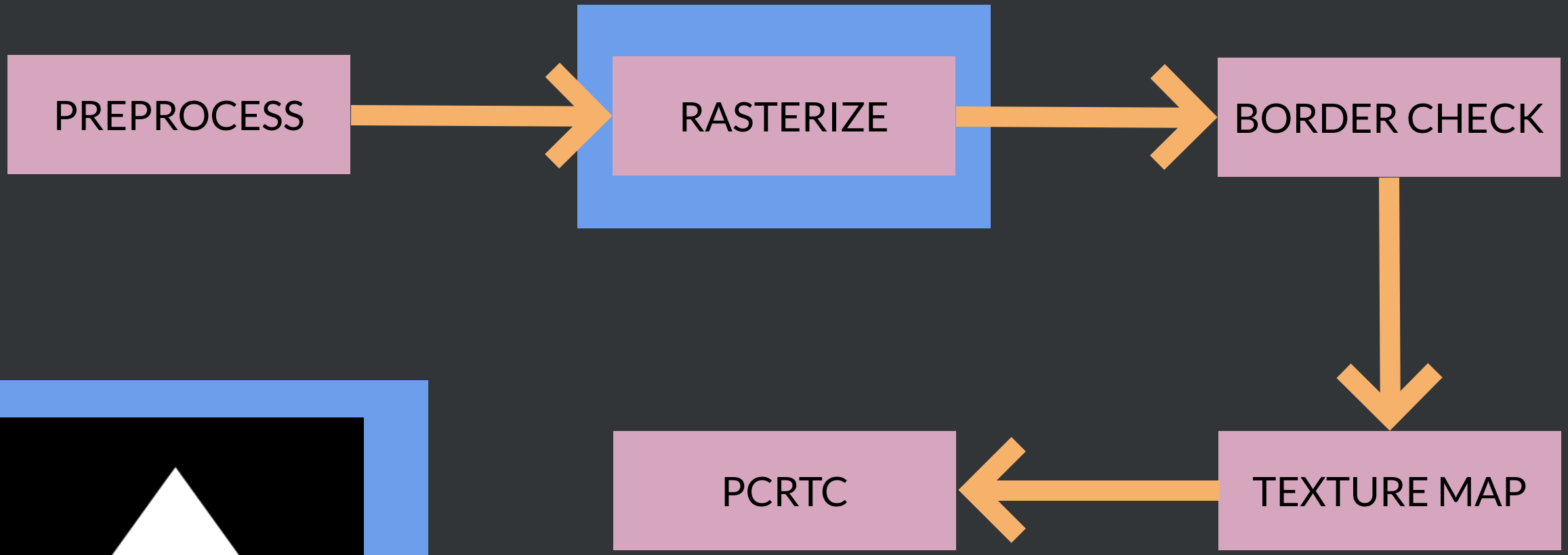
WHAT IS THE GS

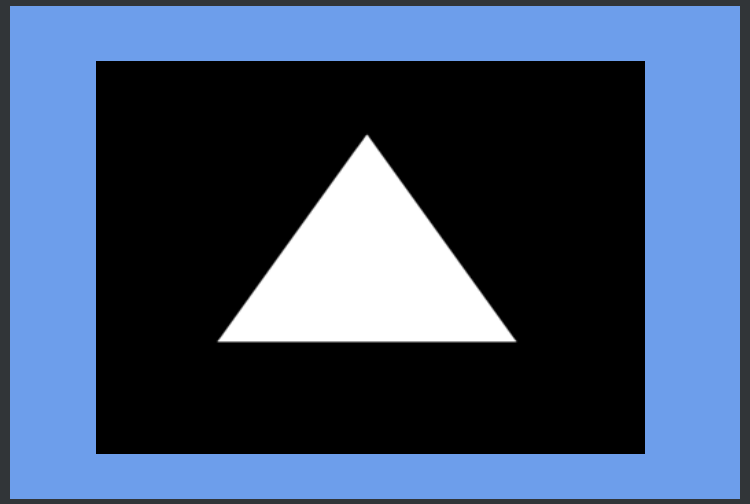


X	Y	Z	W
-1.0	-1.0	0.0	1.0
1.0	-1.0	0.0	1.0
0.0	1.0	0.0	1.0

GS

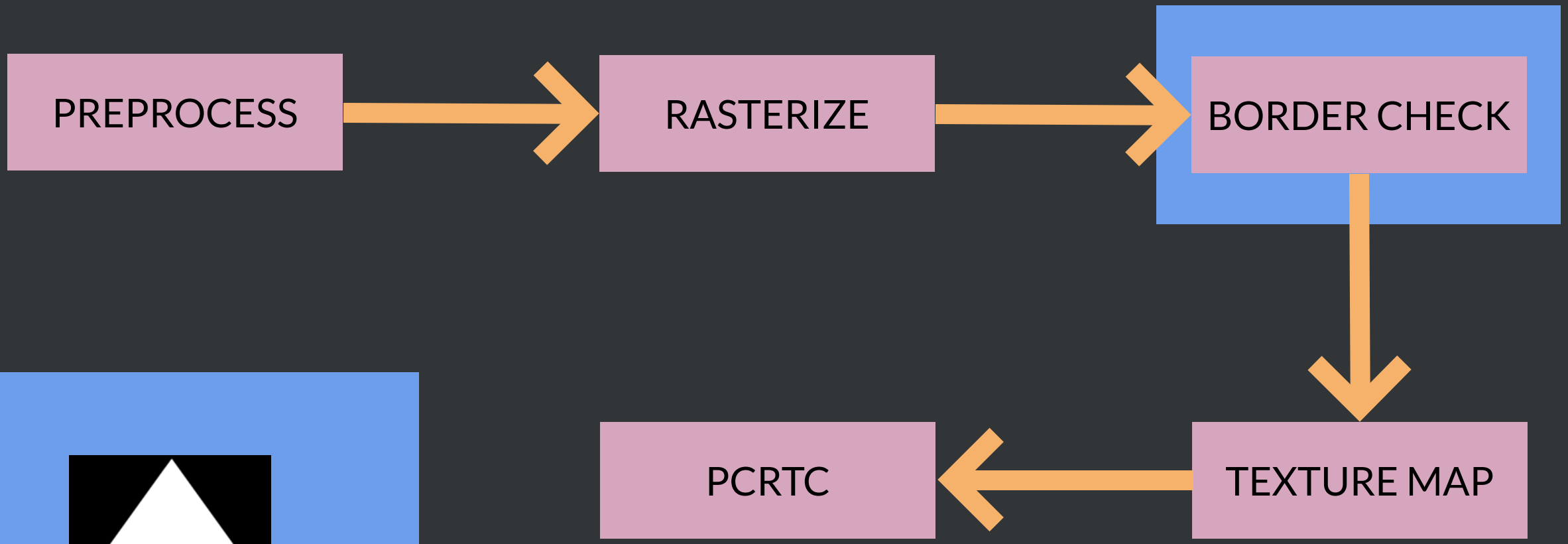
WHAT IS THE GS

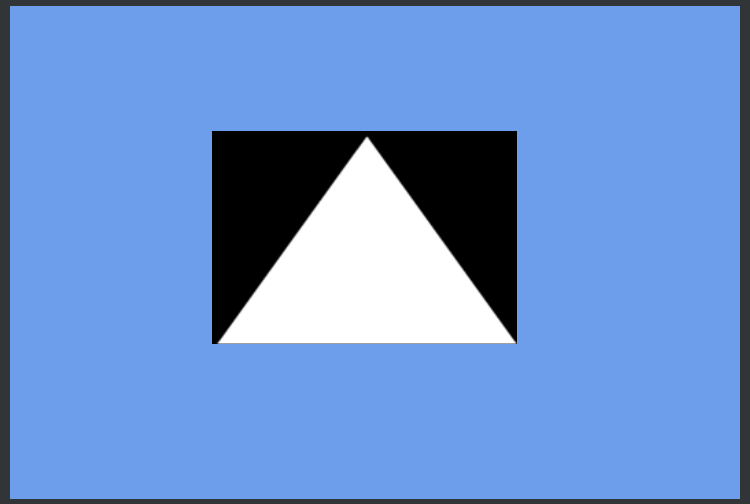




GS

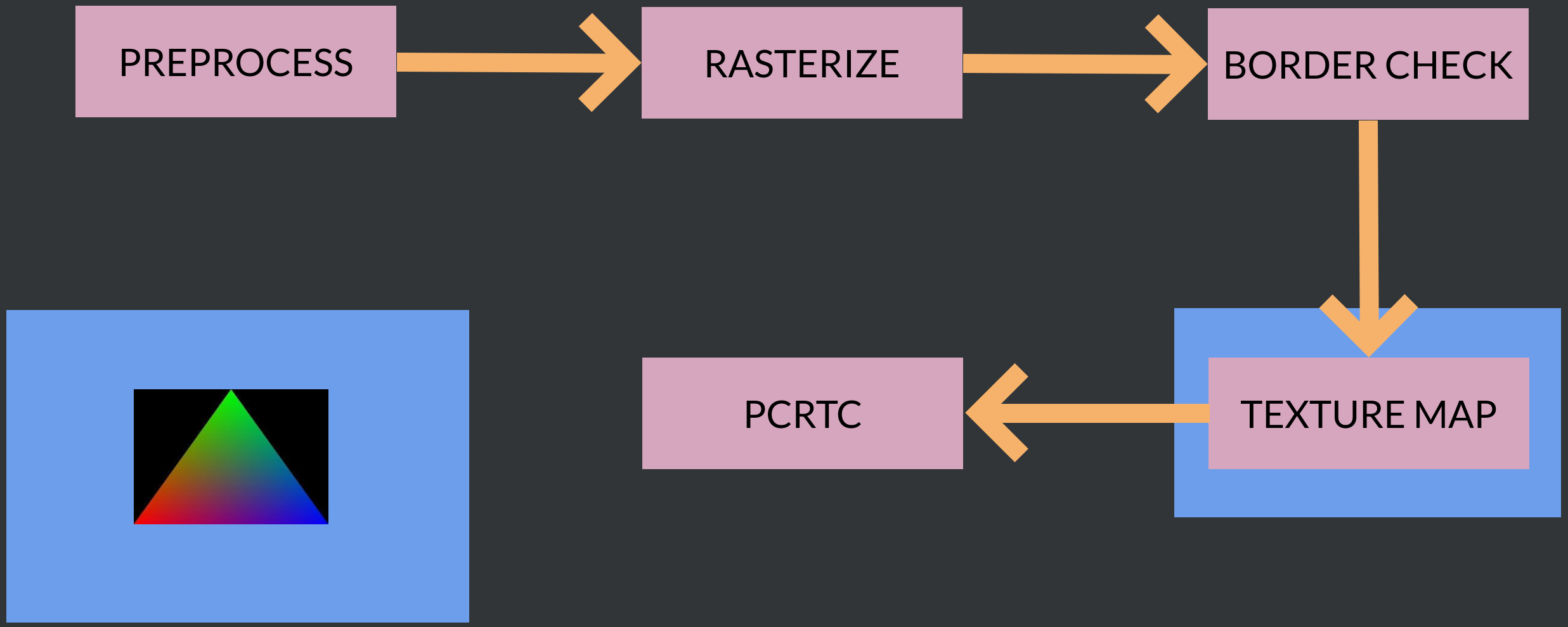
WHAT IS THE GS





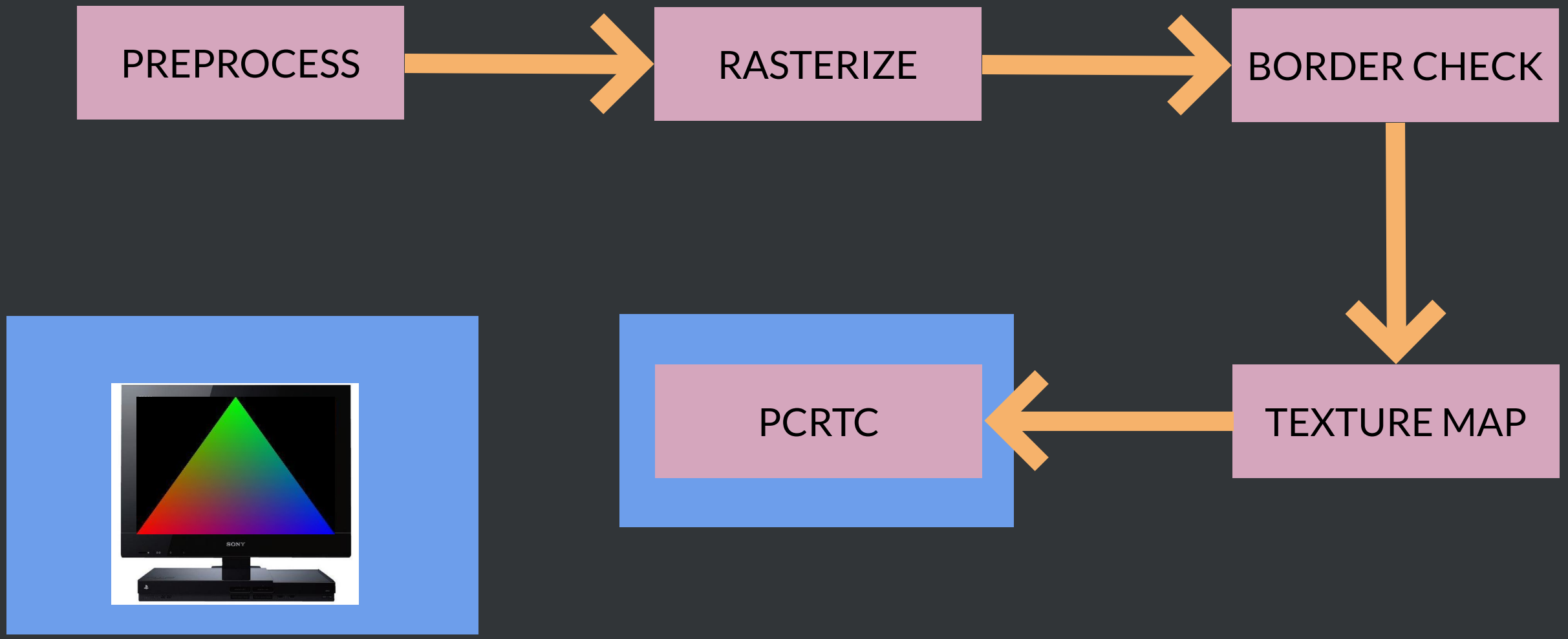
GS

WHAT IS THE GS



GS

WHAT IS THE GS



GS

WHAT IS THE GS

PREPROCESS → RASTERIZE → BORDER CHECK → TEXTURE MAP → PCRTC

And that's a PS2/TV folks!

GS

WHAT IS THE GS

GS

WHAT IS THE GS

Data is transferred to the GS by using the GIF which is a part of the EE

WHAT IS THE GS

Data is transferred to the GS by using the GIF which is a part of the EE

Textures are transferred in a way that pleases the GS pixel units

WHAT IS THE GS

Data is transferred to the GS by using the GIF which is a part of the EE

Textures are transferred in a way that pleases the GS pixel units

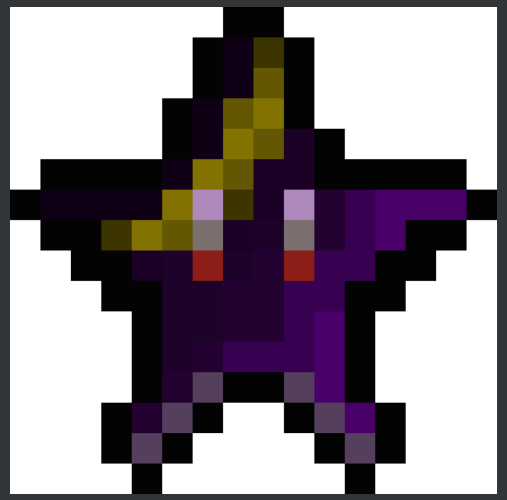
Here is an example with PSMCT32

WHAT IS THE GS

Data is transferred to the GS by using the GIF which is a part of the EE

Textures are transferred in a way that pleases the GS pixel units

Here is an example with PSMCT32



GS

WHAT IS THE GS

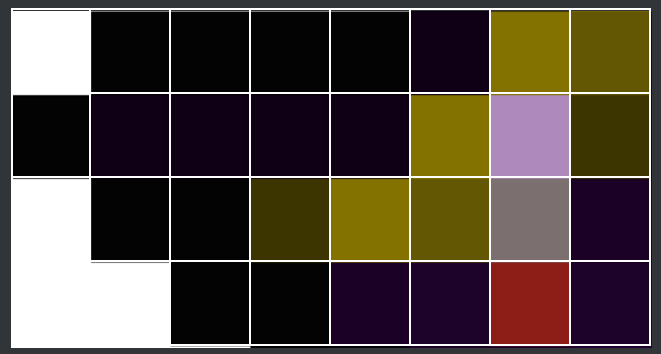
GS

WHAT IS THE GS

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31

GS

WHAT IS THE GS



GS

WHAT IS THE GS

GS

WHAT IS THE GS

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31

0	1	4	5	16	17	20	21
2	3	6	7	18	19	22	23
8	9	12	13	24	25	28	29
10	11	14	15	26	27	30	31

GS

WHAT IS THE GS

GS

WHAT IS THE GS

WHAT IS THE GS

4 and 8 bit textures can optionally be indexed and use a palette, which is called in a PS2 world a CLUT (Color LookUp Table)

WHAT IS THE GS

4 and 8 bit textures can optionally be indexed and use a palette, which is called in a PS2 world a CLUT (Color LookUp Table)

Other notable thing: the Framebuffer doesn't have a fixed size and can be resized to 1080p!

WHAT IS THE GS

4 and 8 bit textures can optionally be indexed and use a palette, which is called in a PS2 world a CLUT (Color LookUp Table)

Other notable thing: the Framebuffer doesn't have a fixed size and can be resized to 1080p!

But we cannot output it unfortunately...

WHAT IS THE GS

4 and 8 bit textures can optionally be indexed and use a palette, which is called in a PS2 world a CLUT (Color LookUp Table)

Other notable thing: the Framebuffer doesn't have a fixed size and can be resized to 1080p!

But we cannot output it unfortunately...

...without hacks :D

OTHER HARDWARE

OTHER HARDWARE

GamePads are handled by the IOP. Usually GamePad state is read at each VSync by the game logic

OTHER HARDWARE

GamePads are handled by the IOP. Usually GamePad state is read at each VSync by the game logic

GamePad communicate with the SIO2 protocol to the PS2, which is an extension of the original PS1 protocol

OTHER HARDWARE

GamePads are handled by the IOP. Usually GamePad state is read at each VSync by the game logic

GamePad communicate with the SIO2 protocol to the PS2, which is an extension of the original PS1 protocol

The IPU is a secondary processor hidden in the EE without any ISA

OTHER HARDWARE

GamePads are handled by the IOP. Usually GamePad state is read at each VSync by the game logic

GamePad communicate with the SIO2 protocol to the PS2, which is an extension of the original PS1 protocol

The IPU is a secondary processor hidden in the EE without any ISA

You write data, through DMA, send the command and it decodes the stream in real time.

OTHER HARDWARE

OTHER HARDWARE

Syscon is a separate processor on the motherboard that handles power management related tasks

We essentially can forget about it emulation wise

OTHER HARDWARE

Syscon is a separate processor on the motherboard that handles power management related tasks

We essentially can forget about it emulation wise

The CDVD subsystem is essentially composed of 3 parts: the laser, a DSP to decode the laser signals and mechacon to ensure DRM

OTHER HARDWARE

Syscon is a separate processor on the motherboard that handles power management related tasks

We essentially can forget about it emulation wise

The CDVD subsystem is essentially composed of 3 parts: the laser, a DSP to decode the laser signals and mechacon to ensure DRM

The BIOS also has the infamous CSS algorithm to decode DVDs, this is handled by the IOP

OTHER HARDWARE

OTHER HARDWARE

USB and IEEE 1394 are connected to IOP's DMA access

OTHER HARDWARE

USB and IEEE 1394 are connected to IOP's DMA access

The protocols are game specific.

OTHER HARDWARE

USB and IEEE 1394 are connected to IOP's DMA access

The protocols are game specific.

The SSBUS is essentially the DMA core of the PS2. The EE, IOP, DEV9, CDVD, etc... are all connected to it.

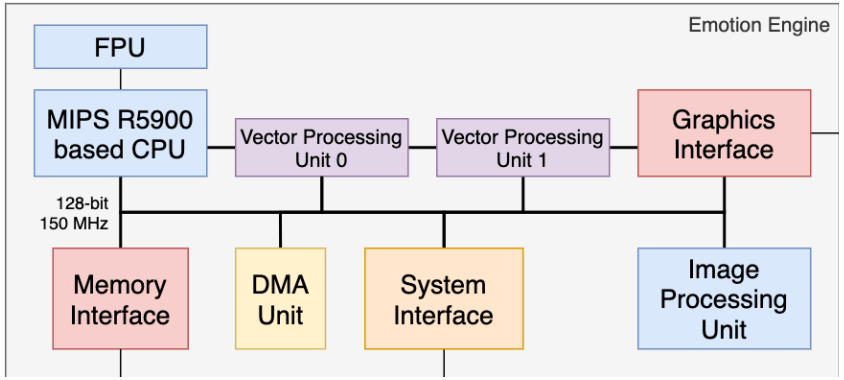
OTHER HARDWARE

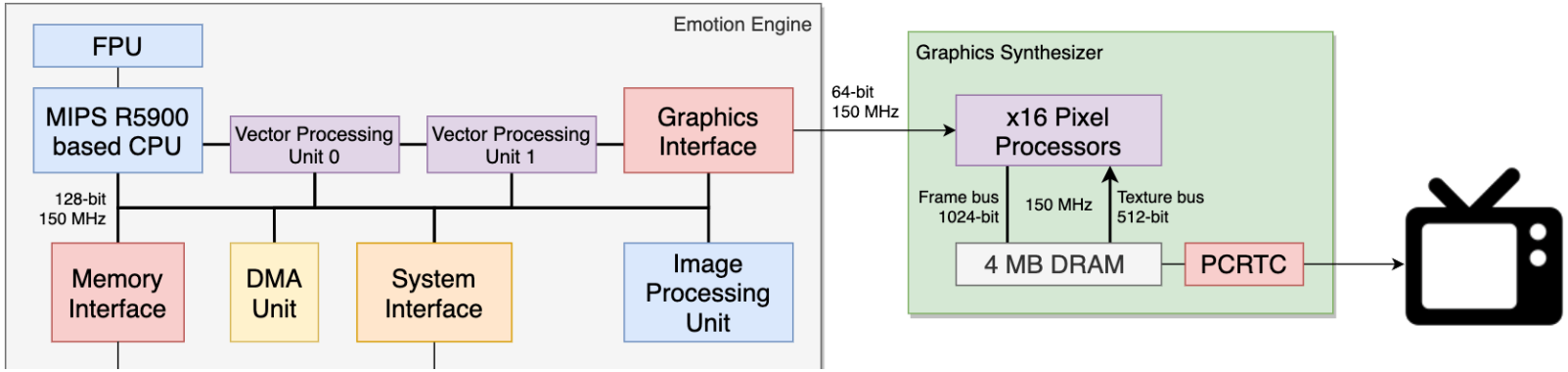
USB and IEEE 1394 are connected to IOP's DMA access

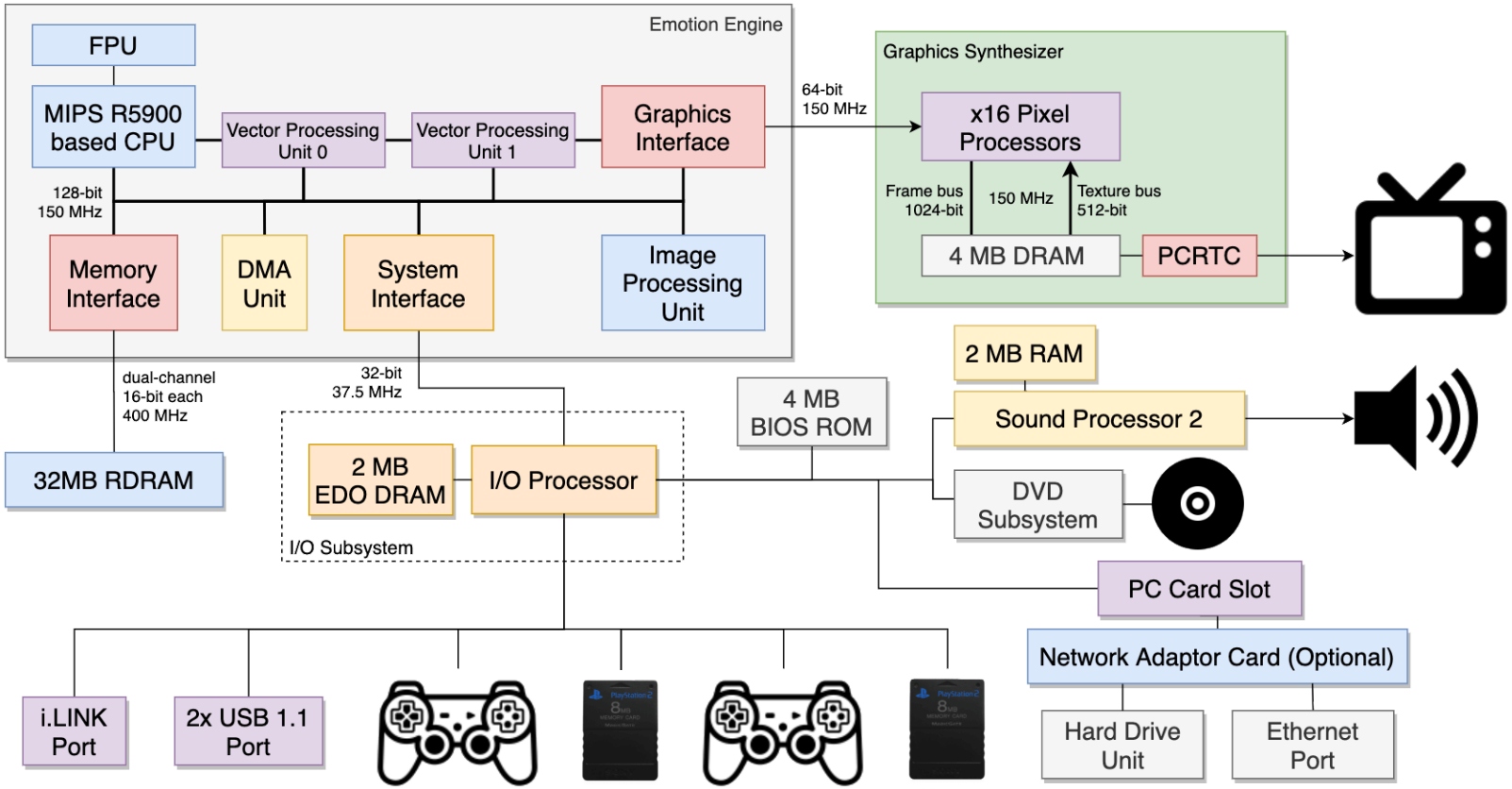
The protocols are game specific.

The SSBUS is essentially the DMA core of the PS2. The EE, IOP, DEV9, CDVD, etc... are all connected to it.

DEV9 is a PCMCIA-like device addressed through DMA. Protocols are game specific but are mostly centered around the ethernet and HDD adapter.


Emotion Engine
FPU
MIPS R5900 based CPU
Vector Processing Unit 0
Vector Processing Unit 1
Graphics Interface
128-bit
150 MHz
Memory Interface
DMA Unit
System Interface
Image Processing Unit

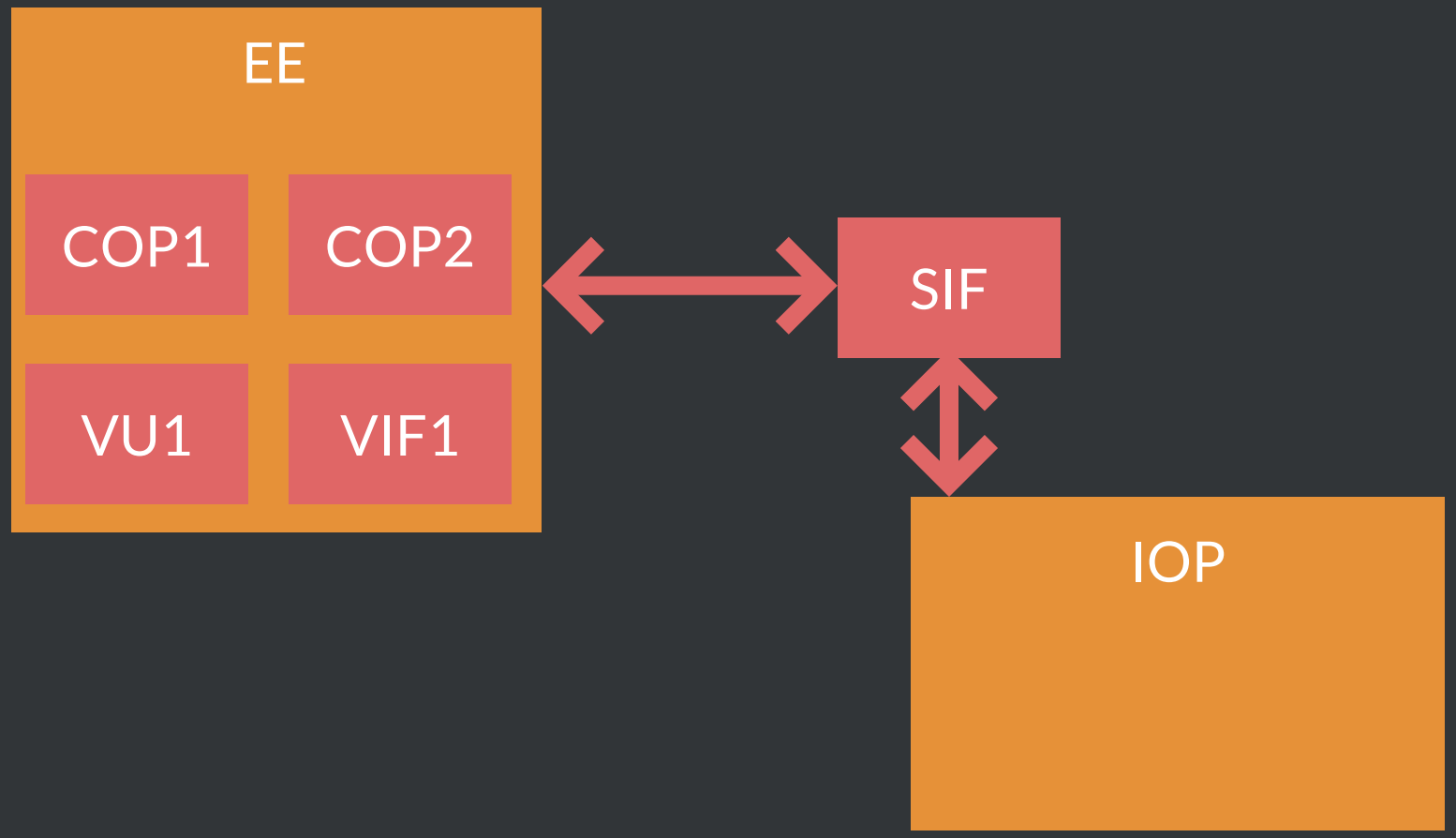

Emotion Engine
FPU
MIPS R5900 based CPU
Vector Processing Unit 0
Vector Processing Unit 1
Graphics Interface
128-bit
150 MHz
Memory Interface
DMA Unit
System Interface
Image Processing Unit
64-bit
150 MHz
Graphics Synthesizer
x16 Pixel Processors
Frame bus
1024-bit
150 MHz
Texture bus
512-bit
4 MB DRAM
PCRTC


Emotion Engine
FPU
MIPS R5900 based CPU
Vector Processing Unit 0
Vector Processing Unit 1
Graphics Interface
128-bit 150 MHz
Memory Interface
DMA Unit
System Interface
Image Processing Unit
64-bit 150 MHz
Graphics Synthesizer
x16 Pixel Processors
Frame bus 1024-bit
150 MHz
Texture bus 512-bit
4 MB DRAM
PCRTC
dual-channel 16-bit each 400 MHz
32MB RDRAM
32-bit 37.5 MHz
2 MB EDO DRAM
I/O Processor
I/O Subsystem
4 MB BIOS ROM
2 MB RAM
Sound Processor 2
DVD Subsystem
PC Card Slot
Network Adaptor Card (Optional)
Hard Drive Unit
Ethernet Port
i.LINK Port
2x USB 1.1 Port

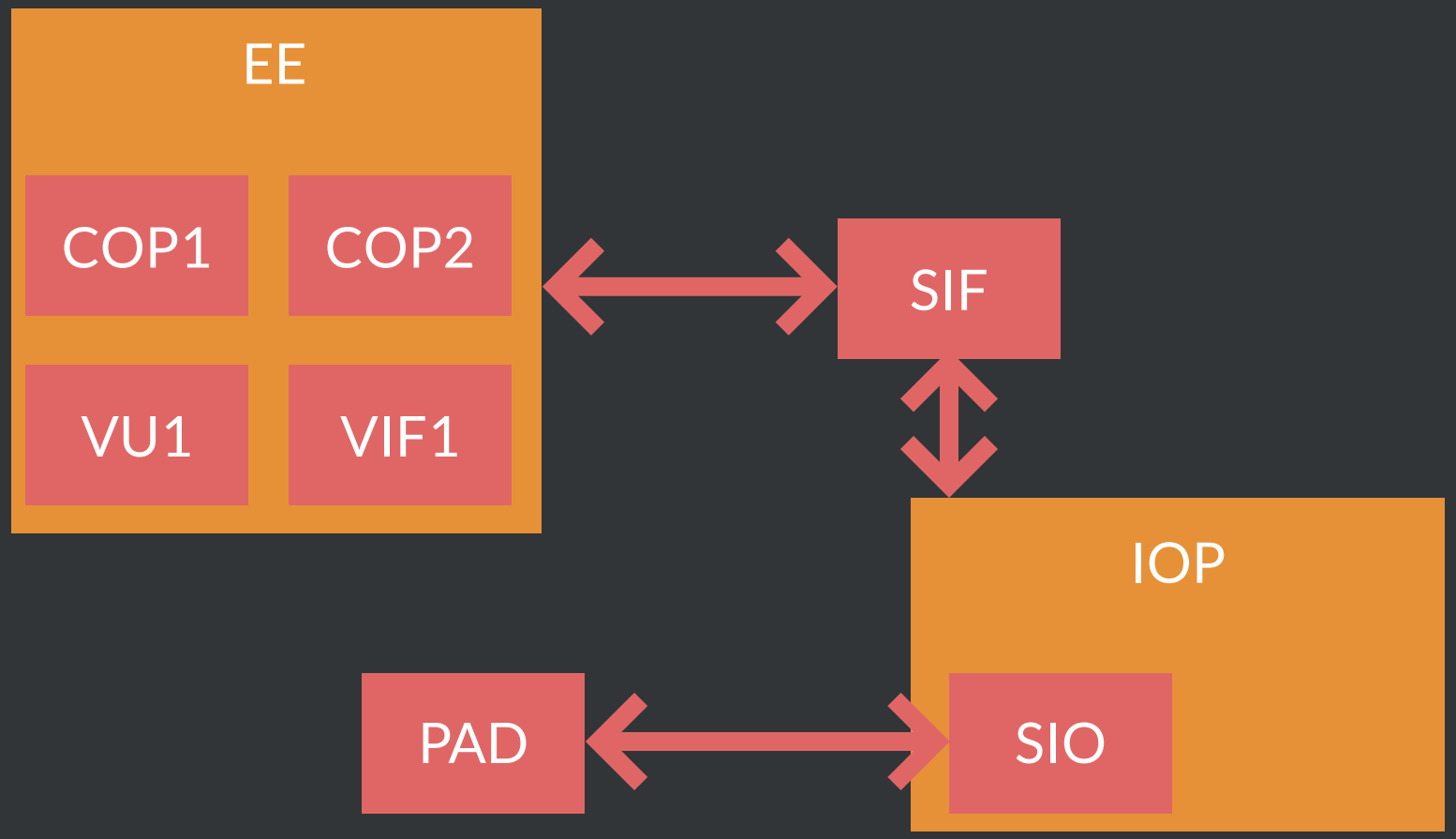
EXAMPLE

EXAMPLE

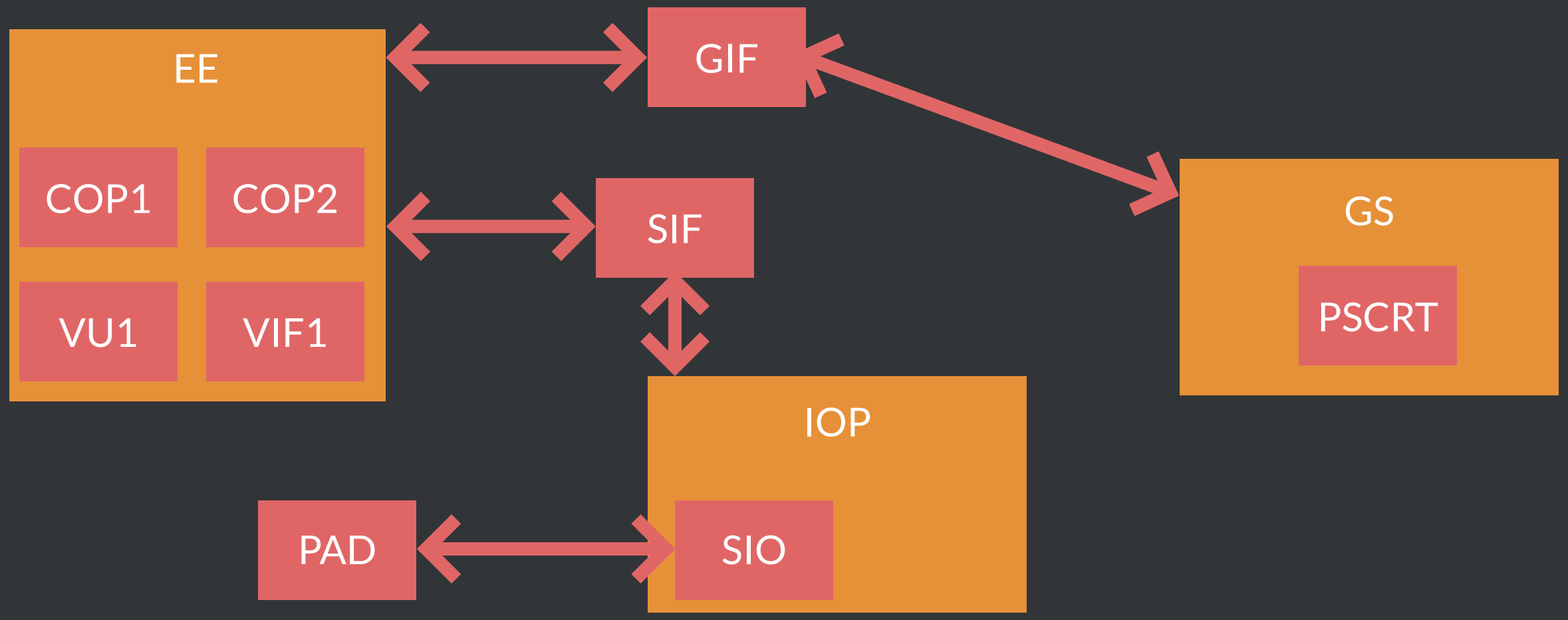
EXAMPLE



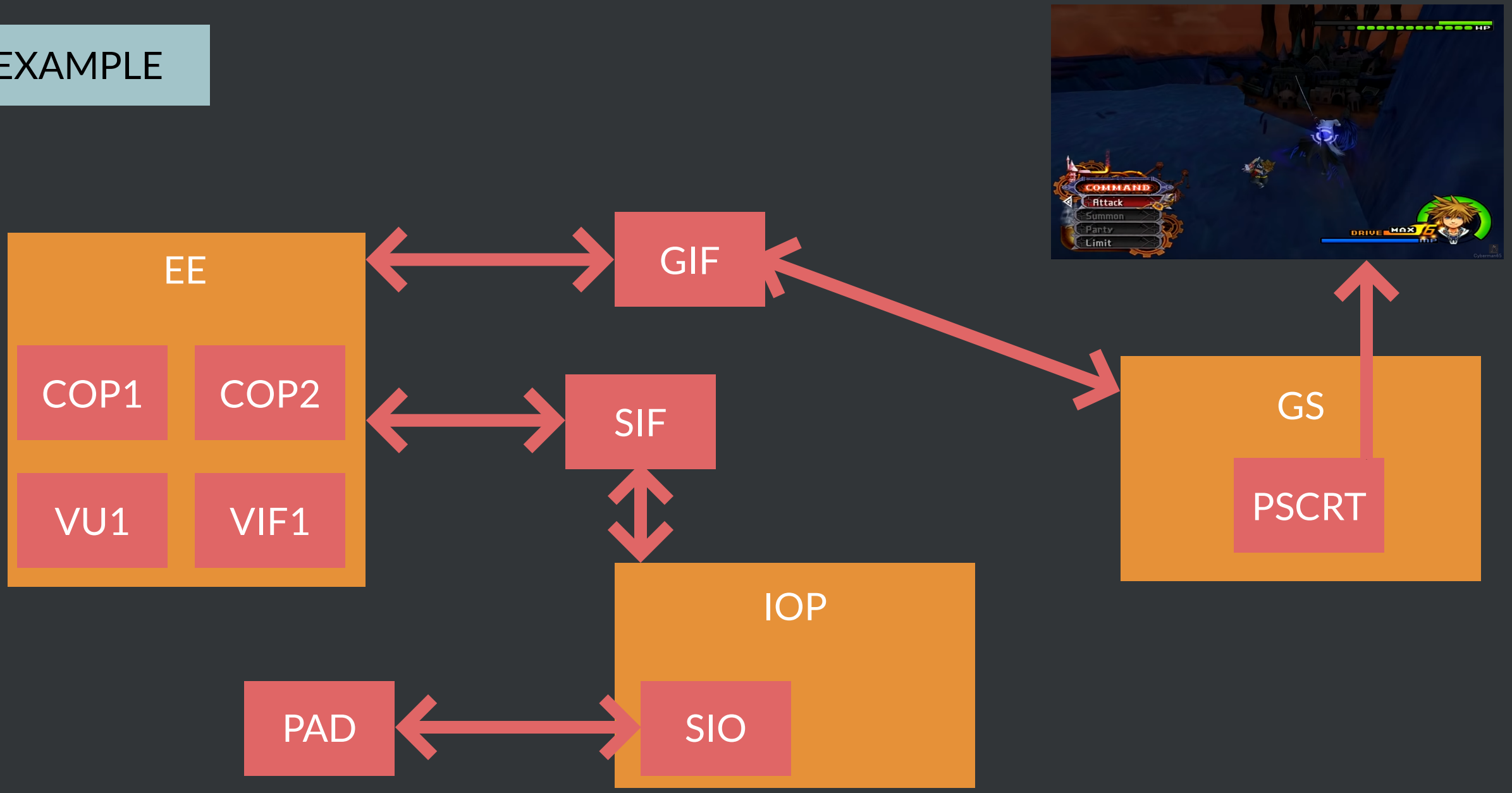
EXAMPLE



EXAMPLE

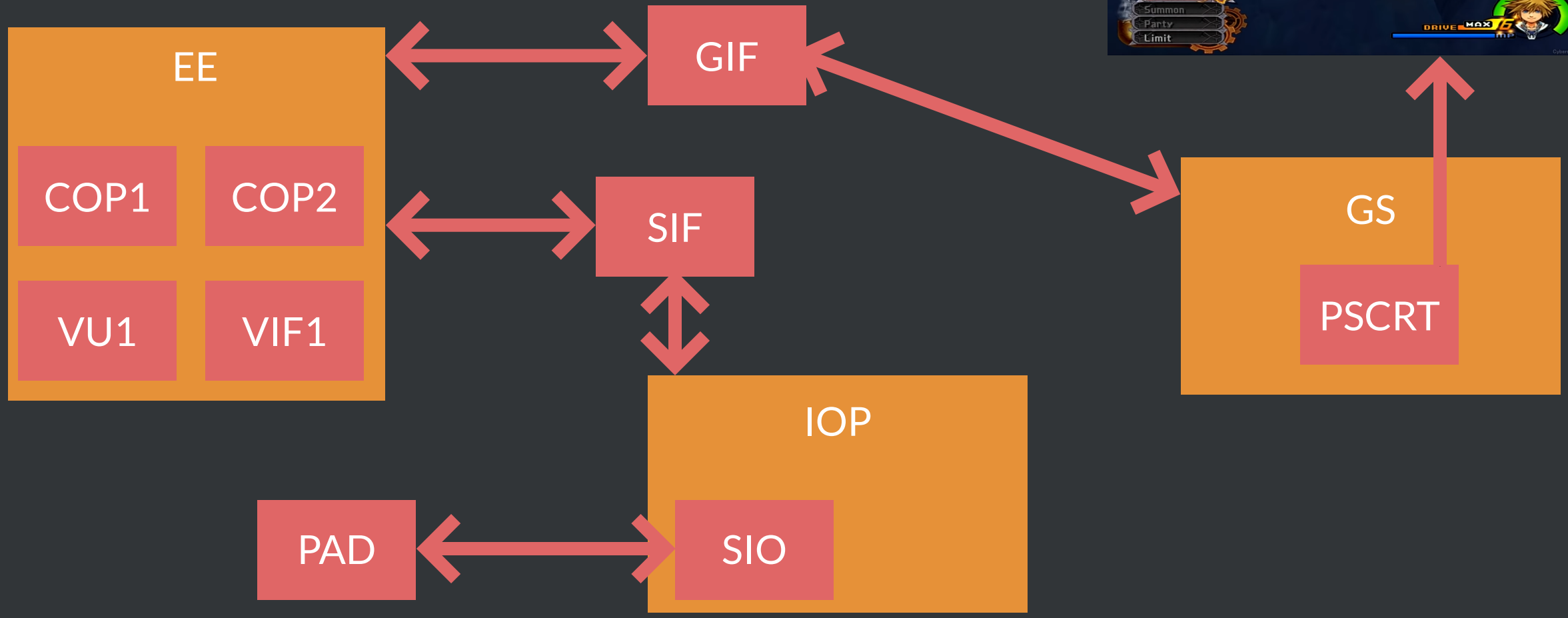


EXAMPLE


EE
COP1
COP2
VU1
VIF1
GIF
SIF
IOP
SIO
PAD
GS
PSCRT
COMMAND
Attack
Summon
Party
Limit
HP
DRIVE
MAX

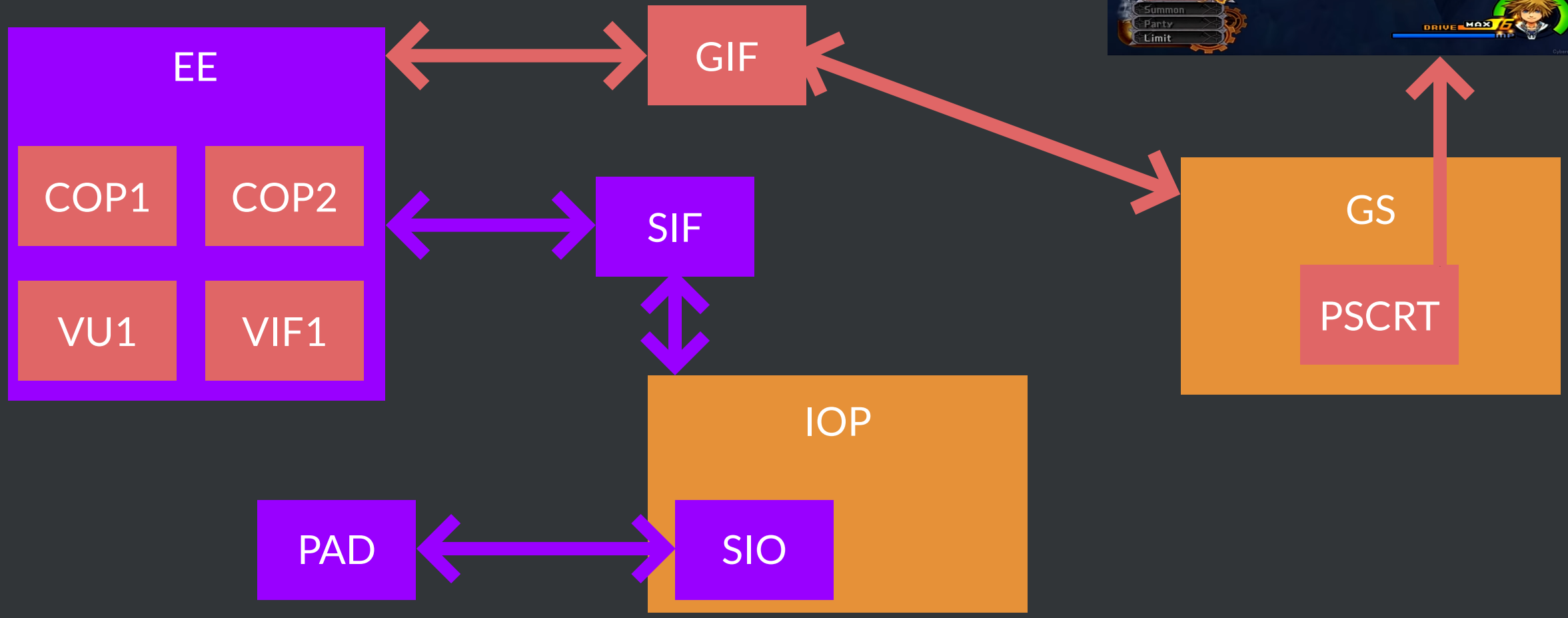
EXAMPLE

A VSync interrupt is reached, EE reads PAD state, transferred through the SIF



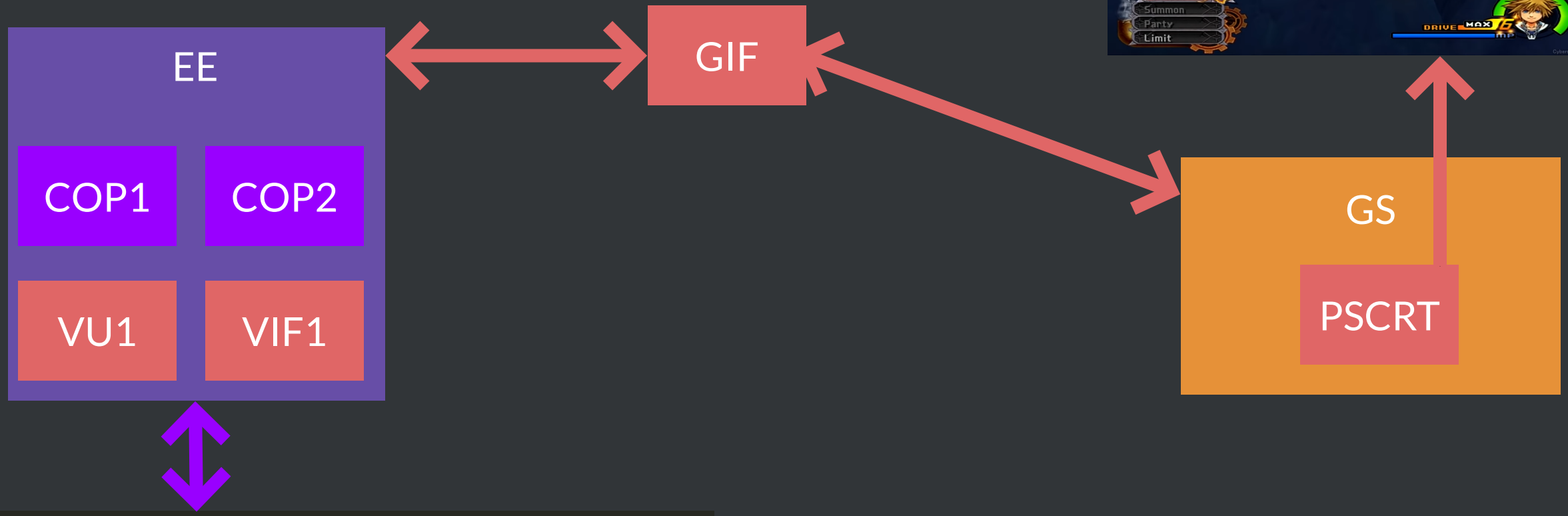
EXAMPLE

A VSync interrupt is reached, EE reads PAD state, transferred through the SIF



EXAMPLE

The EE runs the enemy's AI logic, does some trigonometry for hitbox with COP1 and COP2 for the next frame

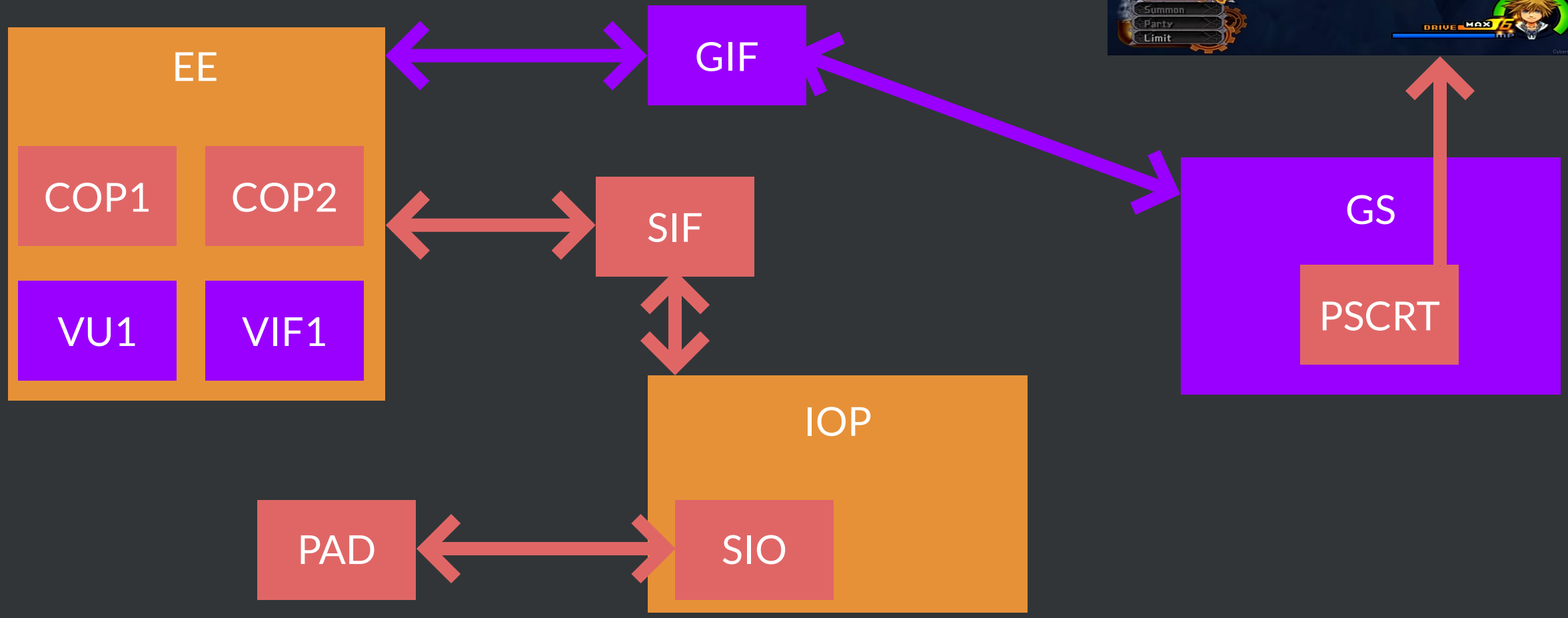



```
# kh2ai pseudocode

delta_hitbox = 0.1
if ((keyblade.hitbox - enemy.hitbox) <= delta_hitbox):
        enemy.attack.one_winged_angel()
```

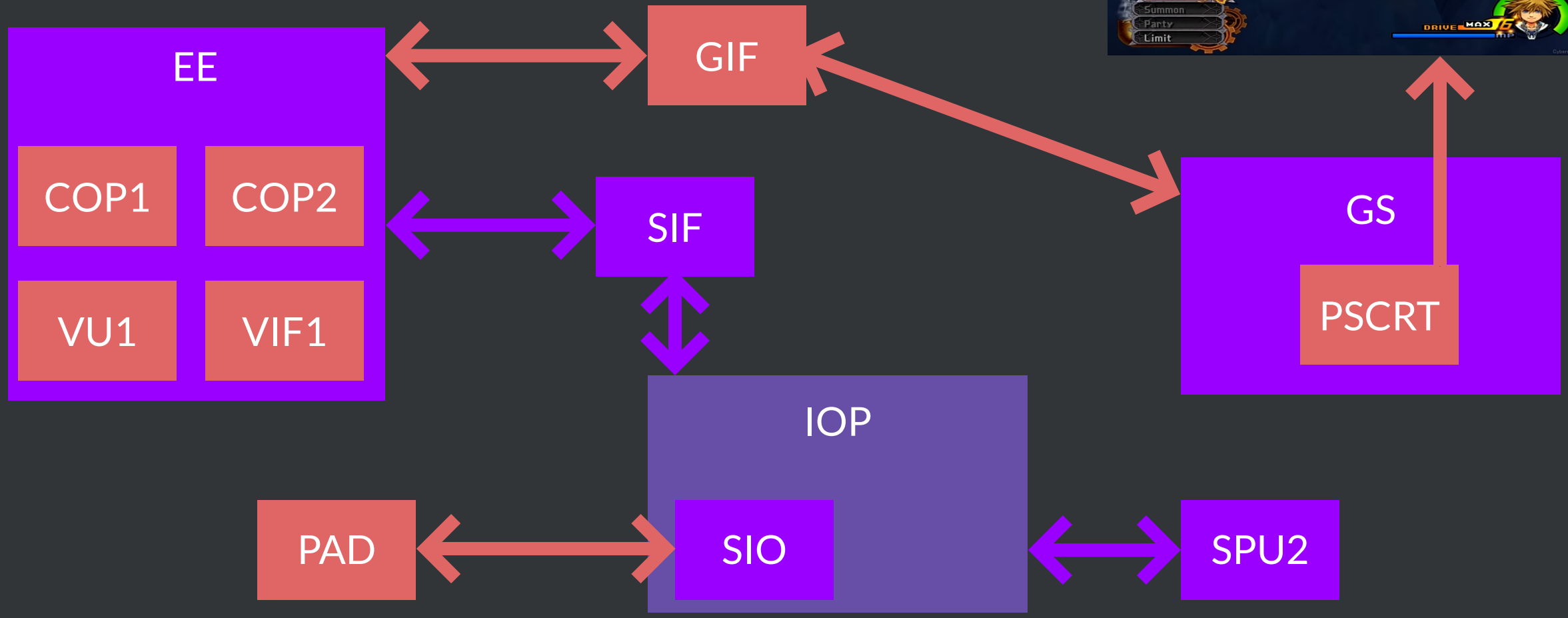
EXAMPLE

Meanwhile the VU1 calculated the transformations of the 3D model for this frame and transfers it to the GS



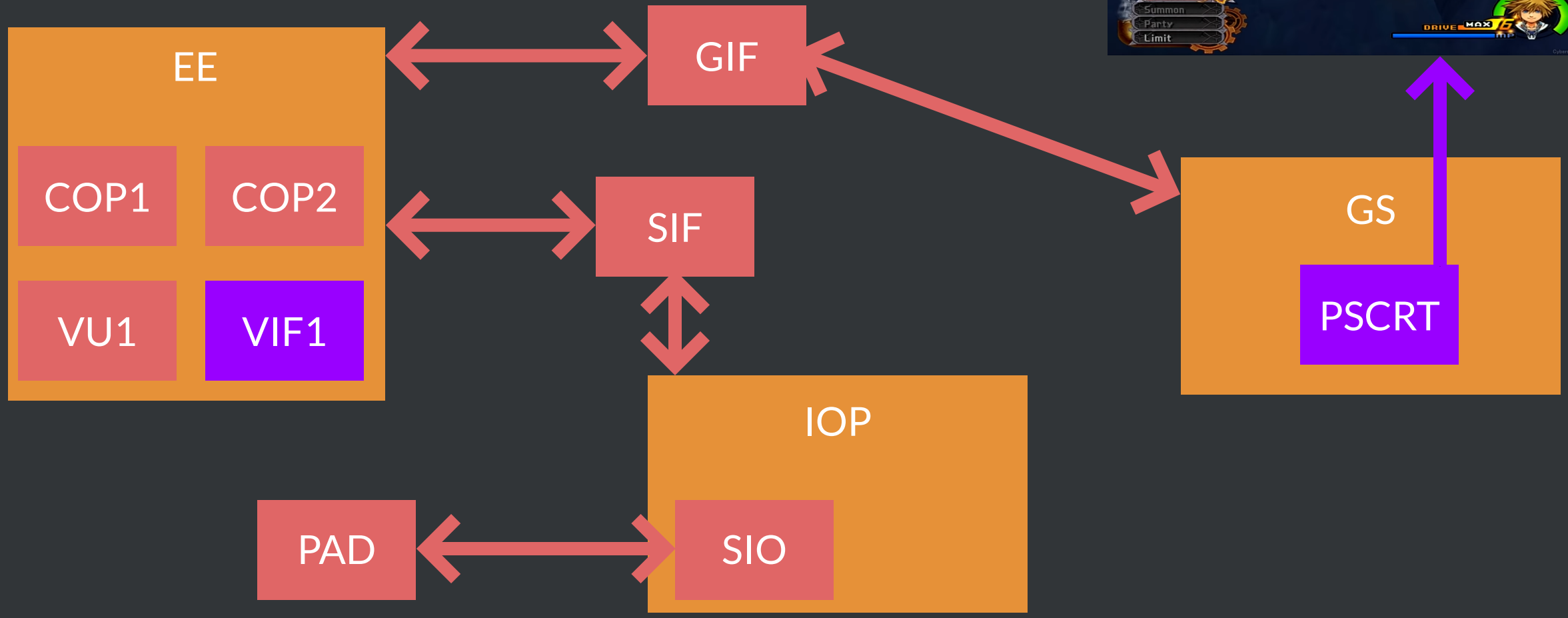
EXAMPLE

The GS is now ready to draw! Meanwhile the game logic continued and need to play a sound effect



EXAMPLE

The GS now draws the frame on screen while the game logic continued and a new 3D model is loaded into VU memory



EXAMPLE

And that's how you get a video game!

EXAMPLE

And that's how you get a video game!

...and diagrams that doesn't make sense

EXAMPLE

And that's how you get a video game!

...and diagrams that doesn't make sense

The core idea is that the game logic, rendering logic and I/O logic are all able to run in parallel on the different cores

EXAMPLE

And that's how you get a video game!

...and diagrams that doesn't make sense

The core idea is that the game logic, rendering logic and I/O logic are all able to run in parallel on the different cores

There is an infinite number of possible arrangements of your rendering pipeline, try to imagine others!

HOW DOES EMULATION WORK

EMULATOR

EMULATOR

What is the first step of making an emulator?

EMULATOR

What is the first step of making an emulator?

```
➜  KH2FM file KH2FM.ISO
KH2FM.ISO: UDF filesystem data (version 1.5) ''
➜  KH2FM file SLPM_666.75
SLPM_666.75: ELF 32-bit LSB executable, MIPS, MIPS-III version 1 (SYSV), statically
linked, stripped
```

EMULATOR

What is the first step of making an emulator?

```
➜  KH2FM file KH2FM.ISO
KH2FM.ISO: UDF filesystem data (version 1.5) ''
➜  KH2FM file SLPM_666.75
SLPM_666.75: ELF 32-bit LSB executable, MIPS, MIPS-III version 1 (SYSV), statically
linked, stripped
```

File parsers!

PARSERS

MEET SYSTEM.CNF

PARSERS

MEET SYSTEM.CNF

```
1  // return value:
2  //   0 - Invalid or unknown disc.
3  //   1 - PS1 CD
4  //   2 - PS2 CD
5  int GetPS2ElfName( wxString& name )
6  {
7          int retype = 0;
8
9          try {
10                 IsoFSCDVD isofs;
11                 IsoFile file( isofs, L"SYSTEM.CNF;1");
12
13                 int size = file.getLength();
14                 if( size == 0 ) return 0;
15     [...]
16 }
```

PARSERS

MEET SYSTEM.CNF

PARSERS

MEET SYSTEM.CNF

```
[...]
while( !file.eof() )
{
        const wxString original( fromUTF8(file.readLine().c_str()) );
        const ParsedAssignmentString parts( original );

        if( parts.lvalue.IsEmpty() && parts.rvalue.IsEmpty() ) continue;
        if( parts.rvalue.IsEmpty() && file.getLength() != file.getSeekPos() )
        { // Some games have a character on the last
          // line of the file, don't print the error in those cases.
                Console.Warning( "(SYSTEM.CNF) Unusual or malformed entry in SYSTEM.CNF ignored:" );
                Console.Indent().WriteLn( original );
                continue;
        }
```

PARSERS

MEET SYSTEM.CNF

PARSERS

MEET SYSTEM.CNF

```
[...]
if( parts.lvalue == L"BOOT2" )
{
        name = parts.rvalue;
        Console.WriteLn( Color_StrongBlue, L"(SYSTEM.CNF) Detected PS2 Disc = " + name );
        retype = 2;
}
else if( parts.lvalue == L"BOOT" )
{
        name = parts.rvalue;
        Console.WriteLn( Color_StrongBlue, L"(SYSTEM.CNF) Detected PSX/PSone Disc = " + name );
        retype = 1;
}
else if( parts.lvalue == L"VMODE" )
{
        Console.WriteLn( Color_Blue, L"(SYSTEM.CNF) Disc region type = " + parts.rvalue );
}
else if( parts.lvalue == L"VER" )
{
        Console.WriteLn( Color_Blue, L"(SYSTEM.CNF) Software version = " + parts.rvalue );
}
```

INTERPRETER

```
1
2
3  void R5900::Interpreter::OpcodeImpl::SWC1() {
4          u32 addr;
5          // force sign extension to 32bit
6          addr = cpuRegs.GPR.r[_Rs_].UL[0] + (s16)(cpuRegs.code & 0xffff);
7          if (addr & 0x00000003)
8          {
9            Console.Error( "FPU (SWC1 Opcode): Invalid Unaligned Memory Address" );
10           return;
11         }  // Should signal an exception?
12         memWrite32(addr, fpuRegs.fpr[_Rt_].UL);
13 }
14
15 void recSWC1()
16 {
17         recCall(::R5900::Interpreter::OpcodeImpl::SWC1);
18 }
```

RECOMPILER

```
1  void recSWC1()
2  {
3  #ifndef FPU_RECOMPILE
4          recCall(::R5900::Interpreter::OpcodeImpl::SWC1);
5  #else
6          _deleteFPtoXMMreg(_Rt_, 1);
7
8          xMOV(arg2regd, ptr32[&fpuRegs.fpr[_Rt_].UL] );
9
10         if( GPR_IS_CONST1( _Rs_ ) )
11         {
12                 int addr = g_cpuConstRegs[_Rs_].UL[0] + _Imm_;
13                 vtlb_DynGenWrite_Const(32, addr);
14         }
15         else
16         {
17                 _eeMoveGPRtoR(arg1regd, _Rs_);
18                 if (_Imm_ != 0)
19                         xADD(arg1regd, _Imm_);
20
21                 iFlushCall(FLUSH_FULLVTLB);
22
23                 vtlb_DynGenWrite(32);
24         }
25
26         EE::Profiler.EmitOp(eeOpcode::SWC1);
27 #endif
28 }
```

SELF-MODIFYING CODE

```
1  void mmap_PageFaultHandler::OnPageFaultEvent( const PageFaultInfo& info, bool& handled )
2  {
3          pxAssert( eeMem );
4
5          // get bad virtual address
6          uptr offset = info.addr - (uptr)eeMem->Main;
7          if( offset >= Ps2MemSize::MainRam ) return;
8
9          mmap_ClearCpuBlock( offset );
10         handled = true;
11 }
12
13 // offset - offset of address relative to psM.
14 // All recompiled blocks belonging to the page are cleared, and any new blocks recompiled
15 // from code residing in this page will use manual protection.
16 static __fi void mmap_ClearCpuBlock( uint offset )
17 {
18         [...]
19 }
```

BIOS

SYSCALL

BIOS

SYSCALL

We can sorta emulate some instructions!

SYSCALL

We can sorta emulate some instructions!

We now need to emulate PS2-specific ones

2 ways to do it:

SYSCALL

We can sorta emulate some instructions!

We now need to emulate PS2-specific ones

2 ways to do it:

- HLE: Reimplement them like an interpreter

SYSCALL

We can sorta emulate some instructions!

We now need to emulate PS2-specific ones

2 ways to do it:

- HLE: Reimplement them like an interpreter
- LLE: Run the BIOS

SYSCALL

We can sorta emulate some instructions!

We now need to emulate PS2-specific ones

2 ways to do it:

- ~~HLE: Reimplement them like an interpreter~~
- LLE: Run the BIOS

SYSCALL

We can sorta emulate some instructions!

We now need to emulate PS2-specific ones

2 ways to do it:

- ~~HLE: Reimplement them like an interpreter~~
- LLE: Run the BIOS

PS2 GAMES PATCH THE BIOS

DUMPING THE BIOS

DUMPING THE BIOS

The BIOS is available on the flash chip!

DUMPING THE BIOS

The BIOS is available on the flash chip!

Unencrypted!!

DUMPING THE BIOS

The BIOS is available on the flash chip!

Unencrypted!!

Save for the DVD EROM, probably to hide the CSS

DUMPING THE BIOS

The BIOS is available on the flash chip!

Unencrypted!!

Save for the DVD EROM, probably to hide the CSS

We don't really care about it though :D

DUMPING THE BIOS

The BIOS is available on the flash chip!

Unencrypted!!

Save for the DVD EROM, probably to hide the CSS

We don't really care about it though :D

A few soldering hackjobs later...

BIOS ENTRYPOINT

```
mfc0        k0,PRId ; get register PRId from COP0
nop
slti        at,k0,0x59 ; if (0x59<=k0) at = 0
bne         at,zero,LAB_00000024 ; if (at == 0) jmp LAB_00000024
nop
```

BIOS ENTRYPOINT

```
mfc0        k0,PRId ; get register PRId from COP0
nop
slti        at,k0,0x59 ; if (0x59<=k0) at = 0
bne         at,zero,LAB_00000024 ; if (at == 0) jmp LAB_00000024
nop
```

BIOS ENTRYPOINT

```
mfc0        k0,PRId ; get register PRId from COP0
nop
slti        at,k0,0x59 ; if (0x59<=k0) at = 0
bne         at,zero,LAB_00000024 ; if (at == 0) jmp LAB_00000024
nop
```

BIOS ENTRYPOINT

```
mfc0        k0,PRId ; get register PRId from COP0
nop
slti        at,k0,0x59 ; if (0x59<=k0) at = 0
bne         at,zero,LAB_00000024 ; if (at == 0) jmp LAB_00000024
nop
```

COP0 is not the same between the IOP and the EE

BIOS ENTRYPOINT

```
mfc0        k0,PRId ; get register PRId from COP0
nop
slti        at,k0,0x59 ; if (0x59<=k0) at = 0
bne         at,zero,LAB_00000024 ; if (at == 0) jmp LAB_00000024
nop
```

COP0 is not the same between the IOP and the EE

This bit of code effectively is the entrypoint for both the IOP and the EE

BIOS ENTRYPOINT

```
mfc0        k0,PRId ; get register PRId from COP0
nop
slti        at,k0,0x59 ; if (0x59<=k0) at = 0
bne         at,zero,LAB_00000024 ; if (at == 0) jmp LAB_00000024
nop
```

COP0 is not the same between the IOP and the EE

This bit of code effectively is the entrypoint for both the IOP and the EE

We already have to emulate the IOP

CUSTOM ARCHITECTURE

CUSTOM ARCHITECTURE

How do you figure out a custom ISA?

CUSTOM ARCHITECTURE

How do you figure out a custom ISA?

Essentially 2 ways:

CUSTOM ARCHITECTURE

How do you figure out a custom ISA?

Essentially 2 ways:

- Make assumptions, test assumptions on hardware

CUSTOM ARCHITECTURE

How do you figure out a custom ISA?

Essentially 2 ways:

- Make assumptions, test assumptions on hardware
- Get documentation from Sony

CUSTOM ARCHITECTURE

How do you figure out a custom ISA?

Essentially 2 ways:

- Make assumptions, test assumptions on hardware
- ~~Get documentation from Sony~~

CUSTOM ARCHITECTURE

How do you figure out a custom ISA?

Essentially 2 ways:

- Make assumptions, test assumptions on hardware
- ~~Get documentation from Sony~~

Here's a talk for some insight on the process:

CUSTOM ARCHITECTURE

How do you figure out a custom ISA?

Essentially 2 ways:

- Make assumptions, test assumptions on hardware
- ~~Get documentation from Sony~~

Here's a talk for some insight on the process:

Reverse engineering of binary programs for custom virtual machines

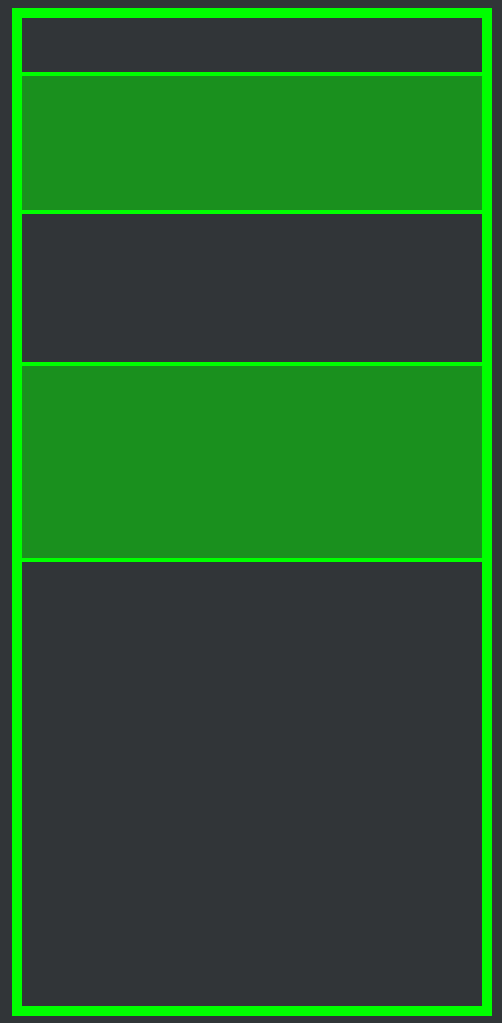
MMU

Memory

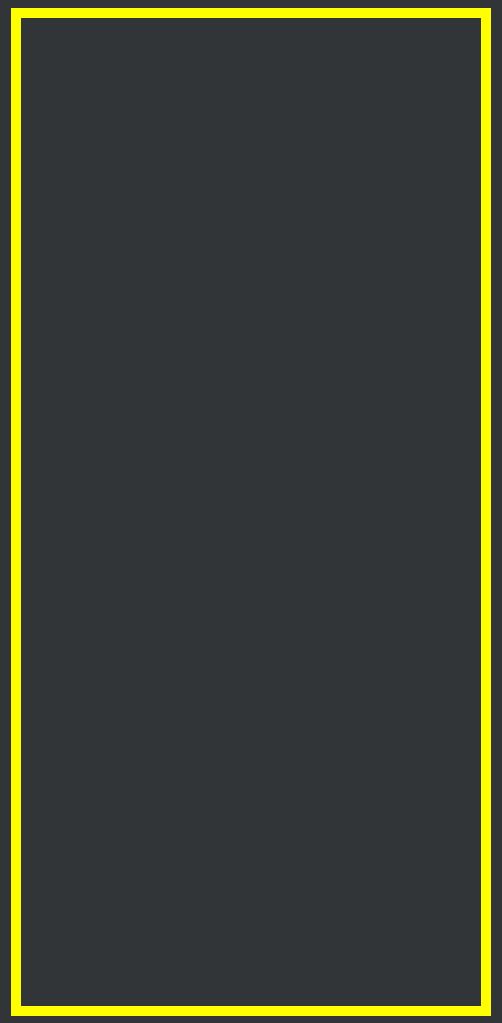
Virtual Memory

MMU

Memory

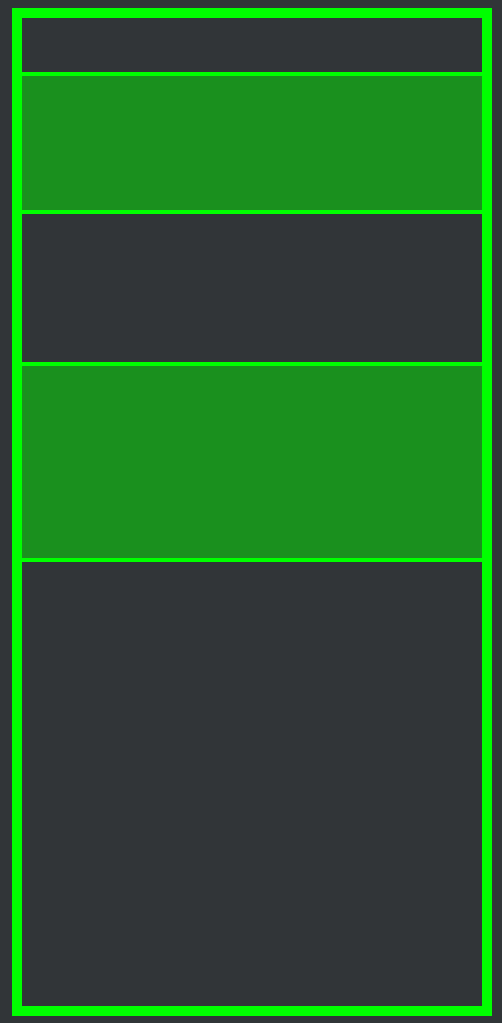


Virtual Memory

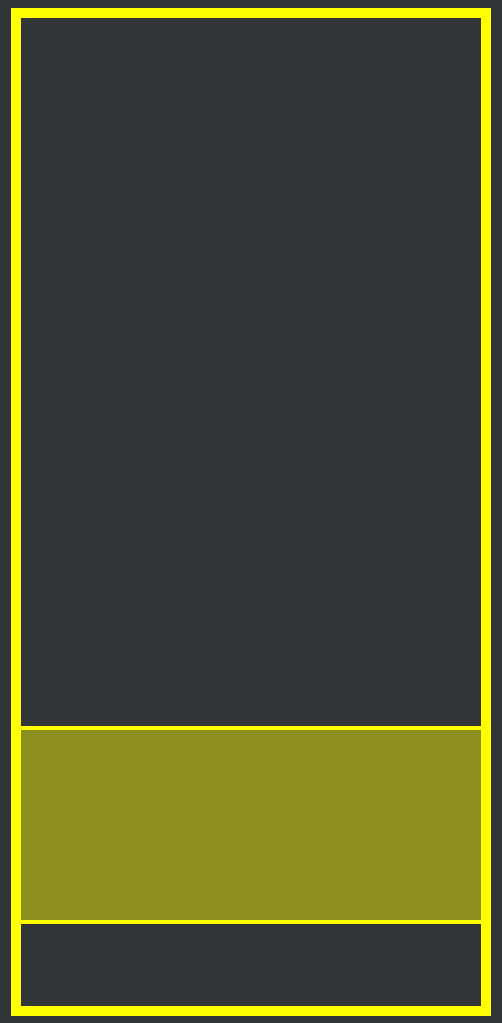


MMU

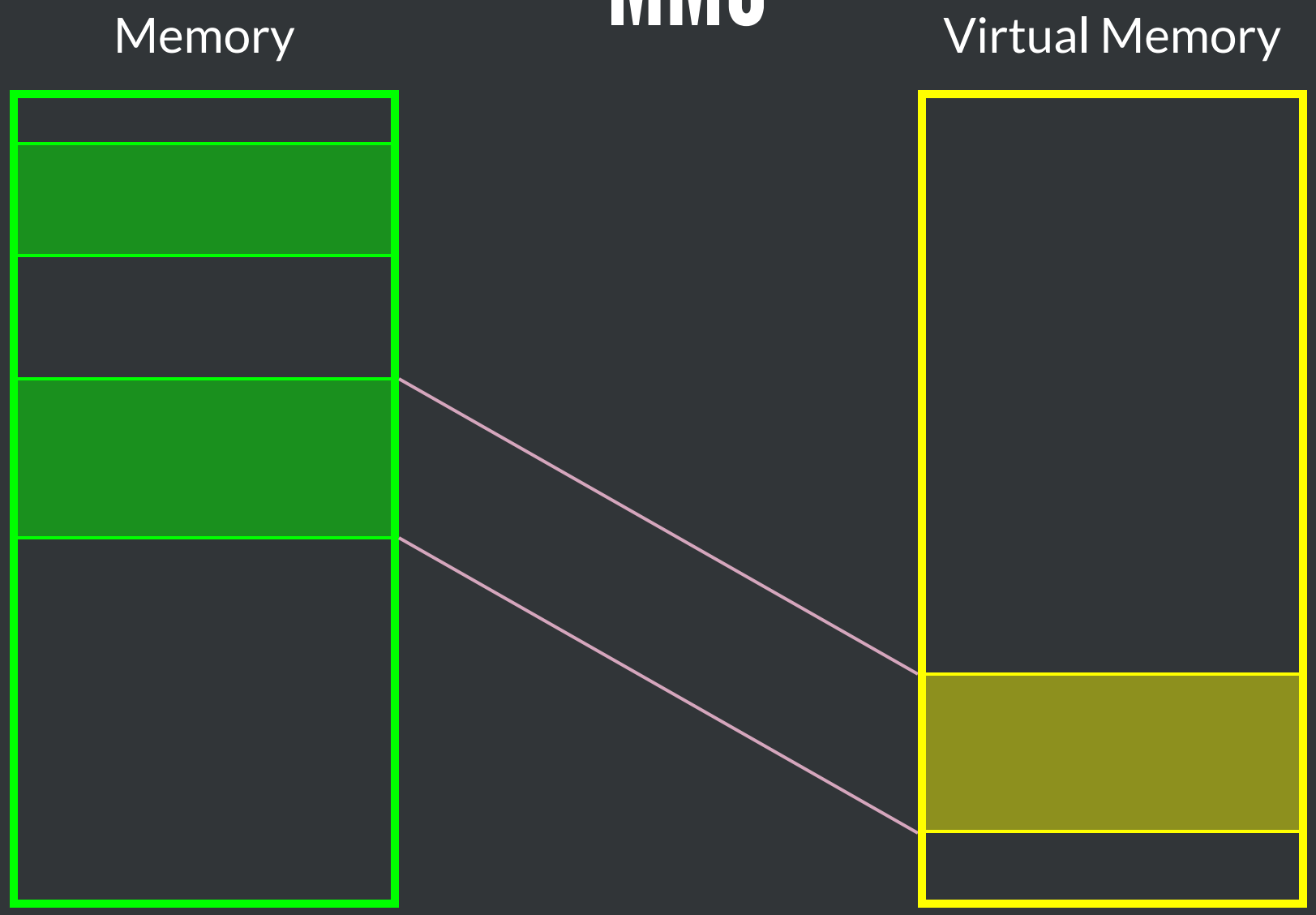
Memory



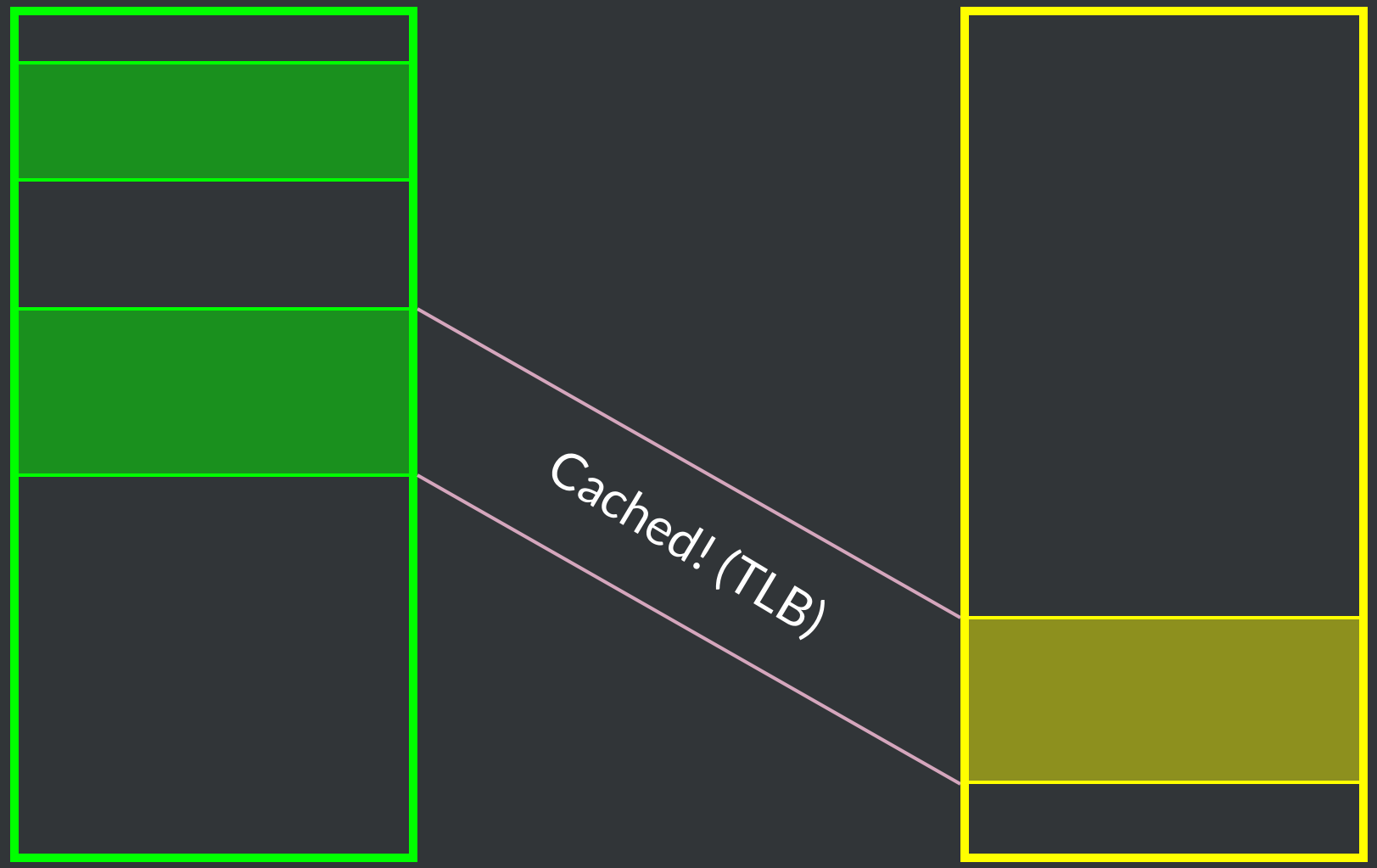
Virtual Memory



MMU



MMU



MMU

The EE has an MMU we need to emulate, meet VTLB!

```
1  void __fastcall vtlb_memRead64(u32 mem, mem64_t *out)
2  {
3          auto vmv = vtlbdata.vmap[mem>>VTLB_PAGE_BITS];
4
5          if (!vmv.isHandler(mem))
6          {
7                  if (!CHECK_EEREC) {
8                          if(CHECK_CACHE && CheckCache(mem))
9                          {
10                                 *out = readCache64(mem);
11                                 return;
12                         }
13                 }
14
15                 *out = *(mem64_t*)vmv.assumePtr(mem);
16 [...]
```

MMU

The EE has an MMU we need to emulate, meet recVTLB!

```
// ---------------------------------------------------------------------------
// TLB lookup is performed in const, with the assumption that the COP0/TLB will clear the
// recompiler if the TLB is changed.
void vtlb_DynGenRead64_Const( u32 bits, u32 addr_const )
{
        EE::Profiler.EmitConstMem(addr_const);

        auto vmv = vtlbdata.vmap[addr_const>>VTLB_PAGE_BITS];
        if( !vmv.isHandler(addr_const) )
        {
                auto ppf = vmv.assumePtr(addr_const);
                switch( bits )
                {
                        case 64:
                                iMOV64_Smart( ptr[arg2reg], ptr[(void*)ppf] );
                        break;

                        case 128:
                                iMOV128_SSE( ptr[arg2reg], ptr[(void*)ppf] );
                        break;

                        jNO_DEFAULT
                }
        }
[...]
```

SysExecutor

MULTI CORE SHENANIGANS

SysExecutor

MULTI CORE SHENANIGANS

Now that we have multiple CPU cores running in parallel we need to handle them concurrently

SysExecutor

MULTI CORE SHENANIGANS

Now that we have multiple CPU cores running in parallel we need to handle them concurrently

We have our own thread scheduler to do that, meet SysExecutor!

SysExecutor

MULTI CORE SHENANIGANS

```
1  void pxEvtQueue::ProcessEvent( SysExecEvent* evt )
2  {
3          if( !evt ) return;
4
5          if( wxThread::GetCurrentId() != m_OwnerThreadId )
6          {
7                  SynchronousActionState sync;
8                  evt->SetSyncState( sync );
9                  PostEvent( evt );
10                 sync.WaitForResult();
11         }
12         else
13         {
14                 std::unique_ptr<SysExecEvent> deleteMe(evt);
15                 deleteMe->_DoInvokeEvent();
16         }
17         [...]
```

Dispatch

DISPATCHING TO PROCESSORS

Dispatch

DISPATCHING TO PROCESSORS

How do we transfer data from, say, the IOP to DEV9?

Dispatch

DISPATCHING TO PROCESSORS

How do we transfer data from, say, the IOP to DEV9?

Our JIT fallbacks to Interpreters and verifies where the write should go!

Dispatch

DISPATCHING TO PROCESSORS

```
1  static void rpsxSB()
2  {
3      _psxDeleteReg(_Rs_, 1);
4      _psxDeleteReg(_Rt_, 1);
5
6      xMOV(arg1regd, ptr32[&psxRegs.GPR.r[_Rs_]]);
7      if (_Imm_) xADD(arg1regd, _Imm_);
8      xMOV( arg2regd, ptr32[&psxRegs.GPR.r[_Rt_]] );
9      xFastCall((void*)iopMemWrite8, arg1regd, arg2regd );
10 }
11
12 void __fastcall iopMemWrite8(u32 mem, u8 value)
13 {
14         mem &= 0x1fffffff;
15         u32 t = mem >> 16;
16         [...]
17         else
18         {
19                 if (!(p != NULL && !(psxRegs.CP0.n.Status & 0x10000) ))
20                 {
21                         if (t == 0x1000)
22                         {
23                                 DEV9write8(mem, value); return;
24                         }
25                         PSXMEM_LOG("err sb %8.8lx = %x", mem, value);
26                 }
27         }
28 }
```

SPU2

EMULATING SOUND

SPU2

EMULATING SOUND

We run an async loop that processes audio while everything else is running

SPU2

EMULATING SOUND

```
1  __forceinline void TimeUpdate(u32 cClocks)
2  {
3          u32 dClocks = cClocks - lClocks;
4
5          // Sanity Checks:
6          //  It's not totally uncommon for the IOP's
7          //  clock to jump backwards a cycle or two, and in
8          //  such cases we just want to ignore the TimeUpdate call.
9          if (dClocks > (u32)-15)
10                 return;
11
12         if (SynchMode == 1) // AsyncMix on
13                 SndBuffer::UpdateTempoChangeAsyncMixing();
14         else
15                 TickInterval = 768; // Reset to default
16
17         //Update Mixing Progress
18         while (dClocks >= TickInterval)
19         {
20                 for (int i = 0; i < 2; i++)
21                 {
22                         if (has_to_call_irq[i])
23                         {
24                                 has_to_call_irq[i] = false;
25                                 if (!(Spdif.Info & (4 << i)) && Cores[i].IRQEnable)
26                                 {
27                                         Spdif.Info |= (4 << i);
28                                         if (!SPU2_dummy_callback)
29                                                 spu2Irq();
30                                 }
31                         }
32                 }
33                 Mix();
34                 [...]
```

SPU2

EMULATING SOUND

```
__forceinline void Mix()
{
        [...]
        Out.Left *= FinalVolume;
        Out.Right *= FinalVolume;

        SndBuffer::Write(Out);
        [...]
}


void SndBuffer::Write(const StereoOut32& Sample)
{
        [...]
        else
        {
                if (SynchMode == 0) // TimeStrech on
                        timeStretchWrite();
                else
                        _WriteSamples(sndTempBuffer, SndOutPacketSize);
        }
}


void SndOut_SDL::callback_fillBuffer(void* userdata, Uint8* stream, int len)
{
        [...]
        for (Uint16 i = 0; i < sdl_samples; i += SndOutPacketSize)
                SndBuffer::ReadSamples(&buffer[i]);
        SDL_MixAudio(stream, (Uint8*)buffer.get(), len, SDL_MIX_MAXVOLUME);
}
```

GS

EMULATING GRAPHICS

```
1  void GSState::FlushPrim()
2  {
3          if (m_index.tail > 0)
4          {
5                  GL_REG("FlushPrim ctxt %d", PRIM->CTXT);
6                  [...]
7                  if (GSLocalMemory::m_psm[m_context->FRAME.PSM].fmt < 3 && GSLocalMemory::m_psm[m_context->ZBUF.PSM].fmt < 3)
8                  {
9                          m_vt.Update(m_vertex.buff, m_index.buff, m_vertex.tail, m_index.tail, GSUtil::GetPrimClass(PRIM->PRIM));
10
11                         m_context->SaveReg();
12
13                         try
14                         {
15                                 Draw();
16                         }
```

GS

EMULATING GRAPHICS

```
1  void GSRendererHW::Draw()
2  {
3          if(m_dev->IsLost() || IsBadFrame()) {
4                  GL_INS("Warning skipping a draw call (%d)", s_n);
5                  return;
6          }
7          GL_PUSH("HW Draw %d", s_n);
8          [...]
9          GSTextureCache::Target* rt = NULL;
10         GSTexture* rt_tex = NULL;
11         if (!no_rt) {
12                 rt = m_tc->LookupTarget(TEX0, m_width, m_height, GSTextureCache::RenderTarget, true, fm);
13                 rt_tex = rt->m_texture;
14         }
15
16         TEX0.TBP0 = context->ZBUF.Block();
17         TEX0.TBW = context->FRAME.FBW;
18         TEX0.PSM = context->ZBUF.PSM;
19
20         GSTextureCache::Target* ds = NULL;
21         GSTexture* ds_tex = NULL;
22         if (!no_ds) {
23                 ds = m_tc->LookupTarget(TEX0, m_width, m_height, GSTextureCache::DepthStencil, context->DepthWrite());
24                 ds_tex = ds->m_texture;
25         }
26         [...]
27         DrawPrims(rt_tex, ds_tex, m_src);
```

GS

EMULATING GRAPHICS

```
1  void GSRendererOGL::DrawPrims(GSTexture* rt, GSTexture* ds, GSTextureCache::Source* tex)
2  {
3          // HLE implementation of the channel selection effect
4          //
5          // Warning it must be done at the begining because it will change the
6          // vertex list (it will interact with PrimitiveOverlap and accurate
7          // blending)
8          EmulateChannelShuffle(&rt, tex);
9
10         // Upscaling hack to avoid various line/grid issues
11         MergeSprite(tex);
12
13         // Always check if primitive overlap as it is used in plenty of effects.
14         m_prim_overlap = PrimitiveOverlap();
15         [...]
16         // Blend
17         if (!IsOpaque() && rt) {
18                 EmulateBlending(DATE_GL42, DATE_GL45);
19         } else {
20                 dev->OMSetBlendState(); // No blending please
21         }
22
23         if (m_ps_sel.dfmt == 1) {
24                 // Disable writing of the alpha channel
25                 m_om_csel.wa = 0;
26         }
27
28         if (DATE && !DATE_GL45) {
29                 GSVector4i dRect = ComputeBoundingBox(rtscale, rtsize);
30         }
31
32         dev->BeginScene();
33
34         EmulateZbuffer(); // will update VS depth mask
```

OTHERS

OTHER COMPONENTS

OTHERS

OTHER COMPONENTS

PAD, DEV9, USB, MCD and CDVD works relatively similarly and as such I won't mention them for brevity sake

OTHERS

OTHER COMPONENTS

PAD, DEV9, USB, MCD and CDVD works relatively similarly and as such I won't mention them for brevity sake

Memory writes are handled by the module, which simulates the I/O.

OTHERS

OTHER COMPONENTS

PAD, DEV9, USB, MCD and CDVD works relatively similarly and as such I won't mention them for brevity sake

Memory writes are handled by the module, which simulates the I/O.

It is then piped to one of multiple system backend. e.g.:

OTHERS

OTHER COMPONENTS

PAD, DEV9, USB, MCD and CDVD works relatively similarly and as such I won't mention them for brevity sake

Memory writes are handled by the module, which simulates the I/O.

It is then piped to one of multiple system backend. e.g.:

- PAD: SDL

OTHERS

OTHER COMPONENTS

PAD, DEV9, USB, MCD and CDVD works relatively similarly and as such I won't mention them for brevity sake

Memory writes are handled by the module, which simulates the I/O.

It is then piped to one of multiple system backend. e.g.:

- PAD: SDL
- DEV9: TAP

OTHERS

OTHER COMPONENTS

PAD, DEV9, USB, MCD and CDVD works relatively similarly and as such I won't mention them for brevity sake

Memory writes are handled by the module, which simulates the I/O.

It is then piped to one of multiple system backend. e.g.:

- PAD: SDL
- DEV9: TAP
- USB-video: V4L

OTHERS

OTHER COMPONENTS

PAD, DEV9, USB, MCD and CDVD works relatively similarly and as such I won't mention them for brevity sake

Memory writes are handled by the module, which simulates the I/O.

It is then piped to one of multiple system backend. e.g.:

- PAD: SDL
- DEV9: TAP
- USB-video: V4L
- MCD: your filesystem!

OTHERS

OTHER COMPONENTS

PAD, DEV9, USB, MCD and CDVD works relatively similarly and as such I won't mention them for brevity sake

Memory writes are handled by the module, which simulates the I/O.

It is then piped to one of multiple system backend. e.g.:

- PAD: SDL
- DEV9: TAP
- USB-video: V4L
- MCD: your filesystem!
- CDVD: <linux/cdrom.h>

WHAT'S LEFT?

WHAT'S LEFT?

Threading the GS and the VU!

WHAT'S LEFT?

Threading the GS and the VU!

Threading the GS is done by waiting for data to be received then have multiple rendering threads in parallel when all transfers are achieved

WHAT'S LEFT?

Threading the GS and the VU!

Threading the GS is done by waiting for data to be received then have multiple rendering threads in parallel when all transfers are achieved

Fairly safe

WHAT'S LEFT?

Threading the GS and the VU!

Threading the GS is done by waiting for data to be received then have multiple rendering threads in parallel when all transfers are achieved

Fairly safe

Threading the VU is much harder and not nearly as safe

WHAT'S LEFT?

Threading the GS and the VU!

Threading the GS is done by waiting for data to be received then have multiple rendering threads in parallel when all transfers are achieved

Fairly safe

Threading the VU is much harder and not nearly as safe

Still considered a SpeedHack, still break things

WHAT'S LEFT?

Threading the GS and the VU!

Threading the GS is done by waiting for data to be received then have multiple rendering threads in parallel when all transfers are achieved

Fairly safe

Threading the VU is much harder and not nearly as safe

Still considered a SpeedHack, still break things

Read up our dev blog about threading VU1 for more infos!

WHAT'S LEFT?

WHAT'S LEFT?

Not going too fast!!

Merged

Make VU run closer in sync with EE, implement Mbit #3593

refractionpcsx2 merged 2 commits into `PCSX2:master` from `kozarovv:VU_cycles` on Aug 29, 2020

Changes by Refraction:
Implement basic cycle counting for COP2 operations, implement COP2 detection while not interlocking, implement Mbit, change drastically cycles required to run every microprogram. Improved flag handling while COP2 update them.

Additionally:
As explained here: Link. Hardware tests proved that VU run at the same speed as EE mips core. So lets set that in pcsx2 where it is possible.

WHAT'S LEFT?

Merged **Make VU run closer in sync with EE, implement Mbit** #3593
refractionpcsx2 merged 2 commits into `PCSX2:master` from `kozarovv:VU_cycles` on Aug 29, 2020

Changes by Refraction:
Implement basic cycle counting for COP2 operations, implement COP2 detection while not interlocking, implement Mbit, change drastically cycles required to run every microprogram. Improved flag handling while COP2 update them.

Additionally:
As explained here: Link. Hardware tests proved that VU run at the same speed as EE mips core. So lets set that in pcsx2 where it is possible.

Fixed games in this commit:

- (VIF) Hitman games - Could have potentially crashed randomly with TLB misses or FIFO errors, no longer happenin
- 24 The game, Primal, Ghosthunter - No longer need patches to get full speed
- Air Rescue Ranger - Textures are now displayed correctly
- Amplitude - SPS on characters fixed
- Gift, Woody Woodpecker, Kaan - Now work full speed
- Lotus Challenge - Cars are no longer bouncy!
- My Street - missing characters now visible, still exhibit a small amount of SPS in microVU0 but perfect in VU0 Int
- Mike Tysons Heavyweight Box - T posing seethrough characters are now whole and animated
- Next Generation Tennis 2003 - No longer need patch to fix SPS
- Nichibeikan Pro Yakyuu: Final League / World Fantasista - Random glitches are gone
- Phase Paradox - Lighting and Camera in cutscenes are fixed
- Rayman 2 Revolution - Random jittering no longer happens
- Sega Superstars Tennis - SPS on hands/feet is now gone
- Tiger woods PGA Tour 2002 - Fixed player stance
- Tony Hawk 4 - Wakeboarding Unleashed demo no longer crash at loading screen (demo need XGKick hack)
- Totally Spies Totally Party! - Bad SPS somewhat fixed - Will require you to set EE Cyclerate + 3 to completely fix.
- Twisted Metal Head-On - Black doors have now proper colors
- Wakeboarding Unleashed - No longer hangs getting to the menu on release builds
- World Series Baseball 2k3 - No longer hang on loading screen (game still have other issues)

Game issues fixed:
fixes #1448 fixes #3252 fixes #3028 fixes #1473 fixes #94

WHAT'S LEFT?

Faster isn't always better !

Merged **Make VU run closer in sync with EE, implement Mbit** #3593
refractionpcsx2 merged 2 commits into `PCSX2:master` from `kozarovv:VU_cycles` on Aug 29, 2020

Changes by Refraction:
Implement basic cycle counting for COP2 operations, implement COP2 detection while not interlocking, implement Mbit, change drastically cycles required to run every microprogram. Improved flag handling while COP2 update them.

Additionally:
As explained here: Link. Hardware tests proved that VU run at the same speed as EE mips core. So lets set that in pcsx2 where it is possible.

Fixed games in this commit:

- (VIF) Hitman games - Could have potentially crashed randomly with TLB misses or FIFO errors, no longer happenin
- 24 The game, Primal, Ghosthunter - No longer need patches to get full speed
- Air Rescue Ranger - Textures are now displayed correctly
- Amplitude - SPS on characters fixed
- Gift, Woody Woodpecker, Kaan - Now work full speed
- Lotus Challenge - Cars are no longer bouncy!
- My Street - missing characters now visible, still exhibit a small amount of SPS in microVU0 but perfect in VU0 Int
- Mike Tysons Heavyweight Box - T posing seethrough characters are now whole and animated
- Next Generation Tennis 2003 - No longer need patch to fix SPS
- Nichibeikan Pro Yakyuu: Final League / World Fantasista - Random glitches are gone
- Phase Paradox - Lighting and Camera in cutscenes are fixed
- Rayman 2 Revolution - Random jittering no longer happens
- Sega Superstars Tennis - SPS on hands/feet is now gone
- Tiger woods PGA Tour 2002 - Fixed player stance
- Tony Hawk 4 - Wakeboarding Unleashed demo no longer crash at loading screen (demo need XGKick hack)
- Totally Spies Totally Party! - Bad SPS somewhat fixed - Will require you to set EE Cyclerate + 3 to completely fix.
- Twisted Metal Head-On - Black doors have now proper colors
- Wakeboarding Unleashed - No longer hangs getting to the menu on release builds
- World Series Baseball 2k3 - No longer hang on loading screen (game still have other issues)

Game issues fixed:
fixes #1448 fixes #3252 fixes #3028 fixes #1473 fixes #94

WHAT'S LEFT?

WHAT'S LEFT?

Emulating the laws of physics

WHAT'S LEFT?

Emulating the laws of physics

No, really

WHAT'S LEFT?

Emulating the laws of physics

No, really

CDVD: Adjust read speed depending on if in inner/outer edge #3877

Edit | Open with

Merged **refractionpcsx2** merged 2 commits into `master` from `cdvd_timing` 22 days ago

Conversation 17 | Commits 2 | Checks 7 | Files changed 3 | +31 −9

refractionpcsx2 commented on Oct 31, 2020 • edited — Member

Fixes Shadowman 2 Second Coming textures
Fixes Arctic Thunder loading problems
Fixes looping music on ONI title screen and skipping dialogues
Fixes Klonoa 2 missing audio
Fixes SPS at the beginning of matches in Next Generation Tennis 2003 (Ronald Garros) - This one surprised me too

Original comments mentioned Silent Hill 2 being starved during videos, but seems fine to me, but if somebody could test further in to the game, that'd be great.

Edit: It does cause the second video (and later ones) to hang, however this isn't being starved of data, I can run the CDVD at 0.3x the normal speed and it works, so there's some sort of annoying timing issue going on. Can be gotten around by setting the EE cycle rate to -1 or enabling the Fast CDVD speedhack

Reviewers
No reviews

Assignees
No one—assign yourself

Labels
CDVD | High Priority

Projects
None yet

WHAT'S LEFT?

Emulating the laws of physics

No, really

```
// Read speed is roughly 37% at lowest and full speed on outer edge.
// I imagine it's more logarithmic than this
// Required for Shadowman to work
// Use SeekToSector as Sector hasn't been updated yet
const float sectorSpeed = (((float)(cdvd.SeekToSector-offset) / numSectors) * 0.63f) + 0.37f;
//DevCon.Warning("Read speed %f sector %d\n", sectorSpeed, cdvd.Sector);
return ((PSXCLK * cdvd.BlockSize) / ((float)(((mode == MODE_CDROM) ?
        PSX_CD_READSPEED : PSX_DVD_READSPEED) * cdvd.Speed) * sectorSpeed));
```

WHAT'S LEFT?

WHAT'S LEFT?

Making an infrastructure!

WHAT'S LEFT?

Making an infrastructure!

A website, forum, compatibility list, get testers...

WHAT'S LEFT?

Making an infrastructure!

A website, forum, compatibility list, get testers...

This is where **YOU** come in :D

WHAT'S LEFT?

Making an infrastructure!

A website, forum, compatibility list, get testers...

This is where **YOU** come in :D

We always need help, feel free to hang out and say hi!

WHAT'S LEFT?

Making an infrastructure!

A website, forum, compatibility list, get testers...

This is where **YOU** come in :D

We always need help, feel free to hang out and say hi!

https://discord.com/invite/TCz3t9k

WHAT'S LEFT?

Making an infrastructure!

A website, forum, compatibility list, get testers...

This is where **YOU** come in :D

We always need help, feel free to hang out and say hi!

https://discord.com/invite/TCz3t9k

You can bridge it to matrix with
https://github.com/matrix-discord/mx-puppet-discord

STATE

STATE OF THE PROJECT

STATE

STATE OF THE PROJECT

PCSX2 is **really old**

STATE

STATE OF THE PROJECT

PCSX2 is **really old**

It has now a lot of legacy code that simply needs to be redone, redesigned or freshened up

STATE

STATE OF THE PROJECT

PCSX2 is **really old**

It has now a lot of legacy code that simply needs to be redone, redesigned or freshened up

I am leading a whole codebase redesign effort

STATE

STATE OF THE PROJECT

PCSX2 is **really old**

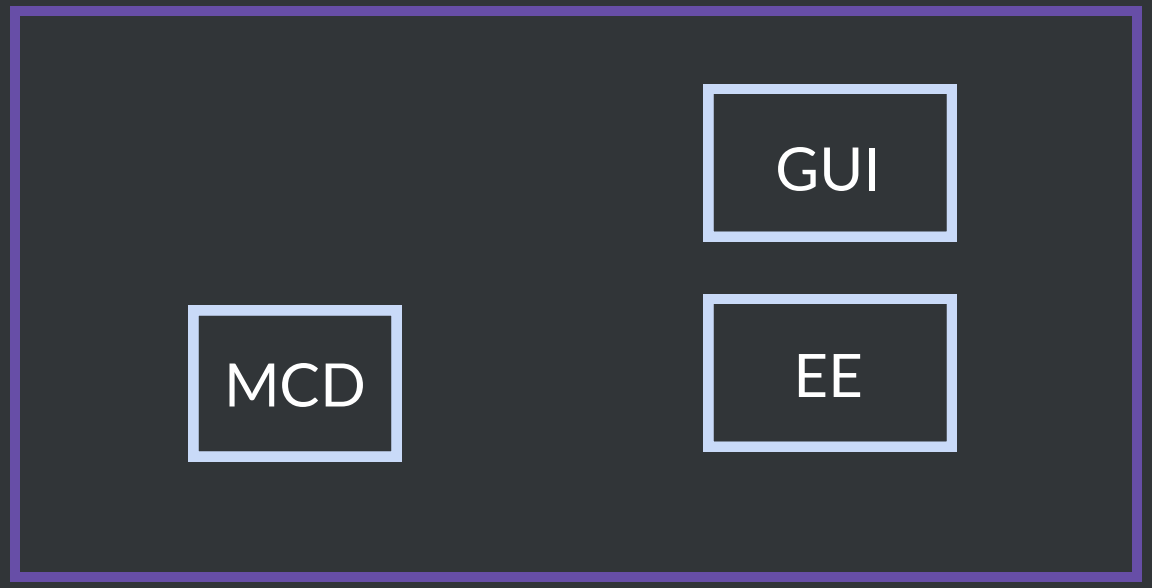
It has now a lot of legacy code that simply needs to be redone, redesigned or freshened up

I am leading a whole codebase redesign effort

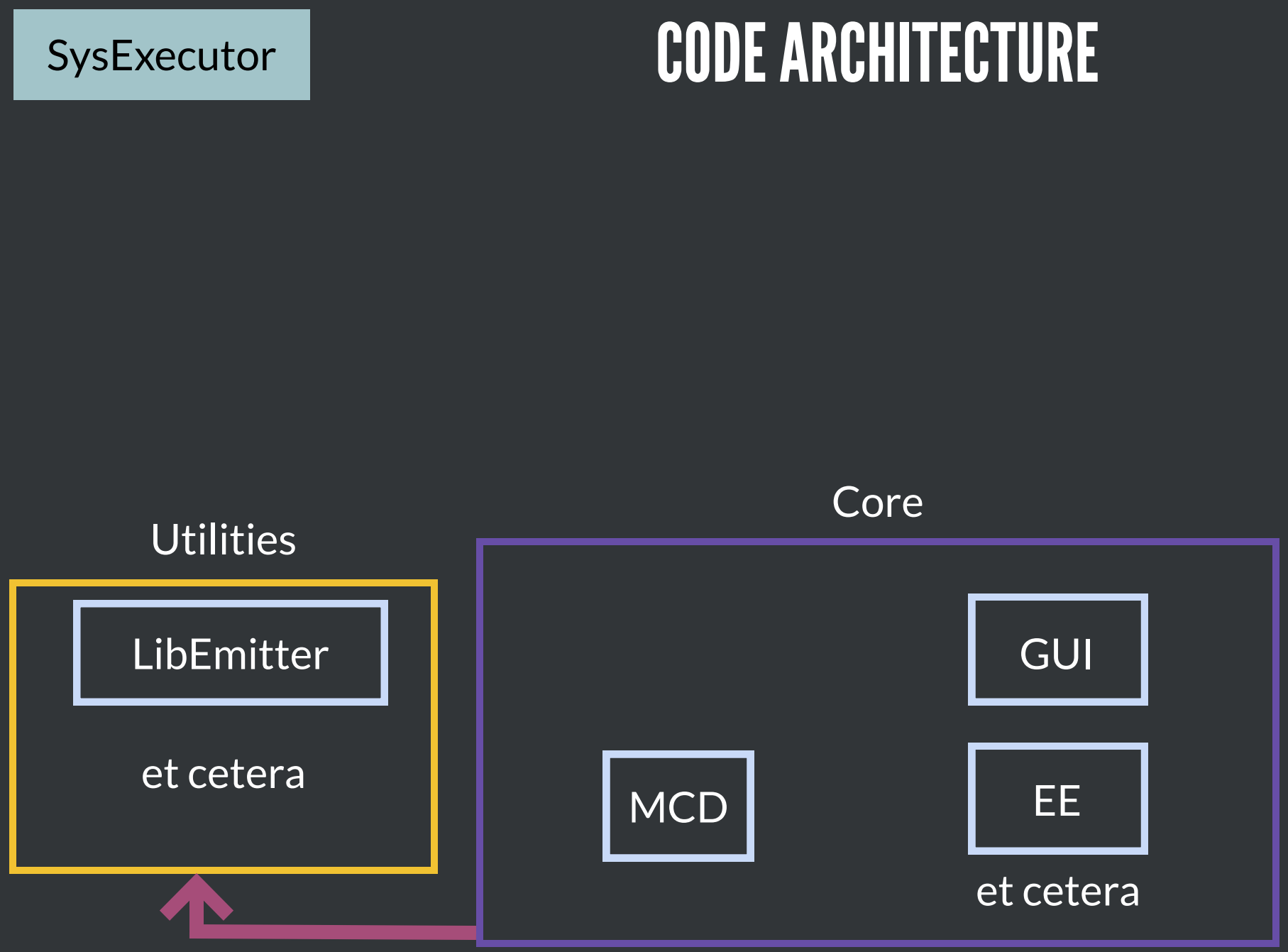
I'll show you in the next slides the state of things and what to expect!

SysExecutor

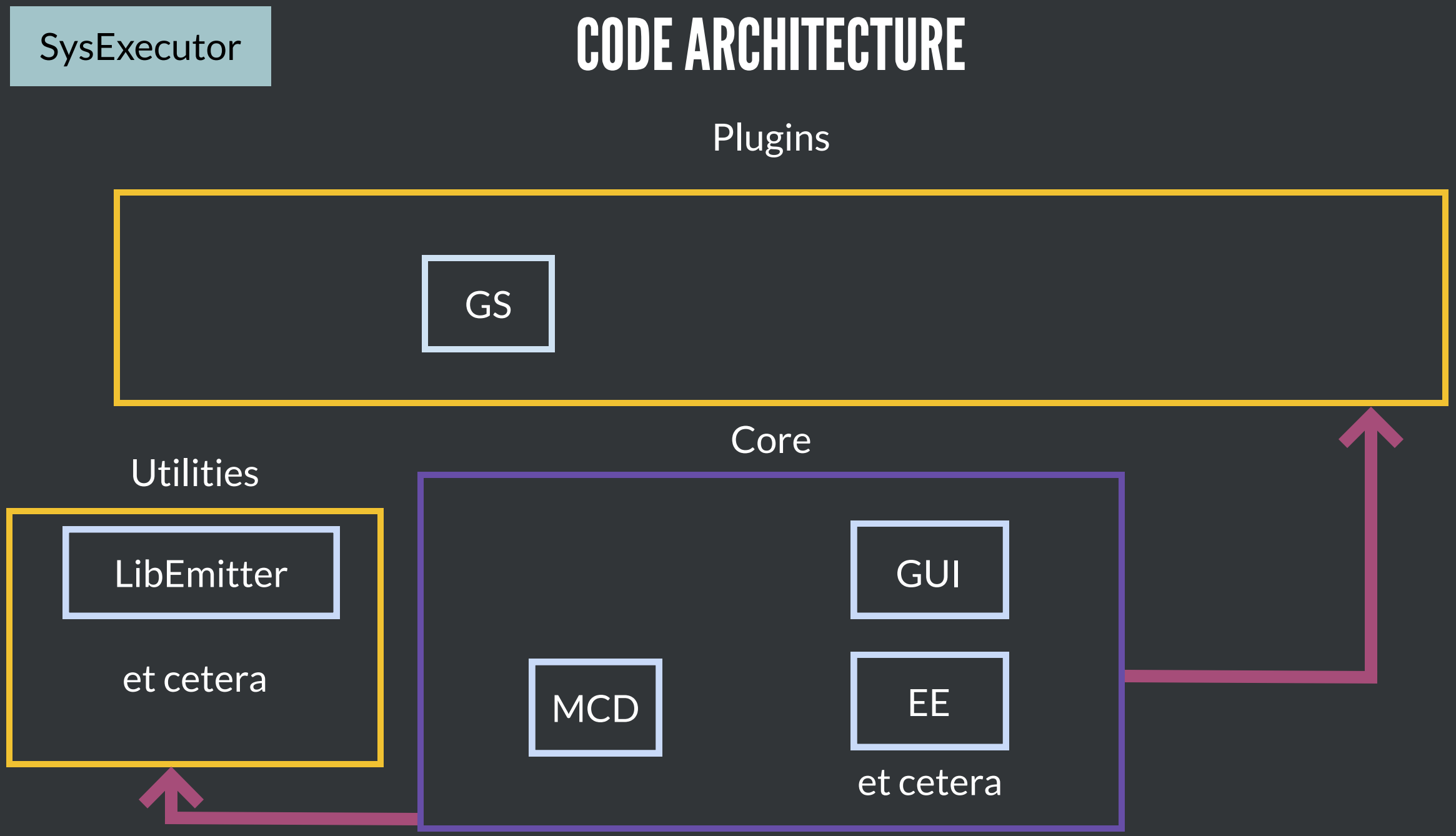
CODE ARCHITECTURE



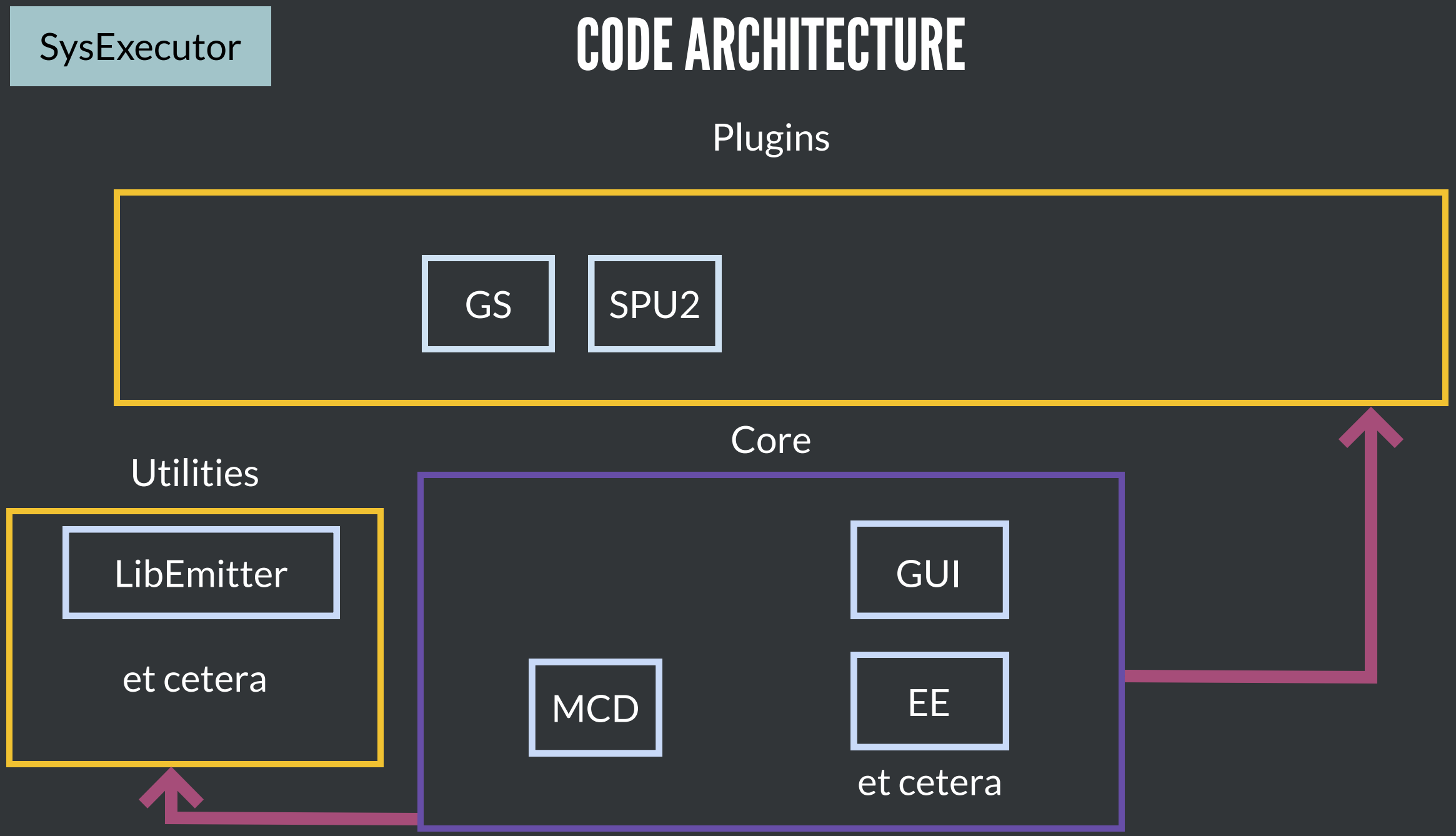
CODE ARCHITECTURE



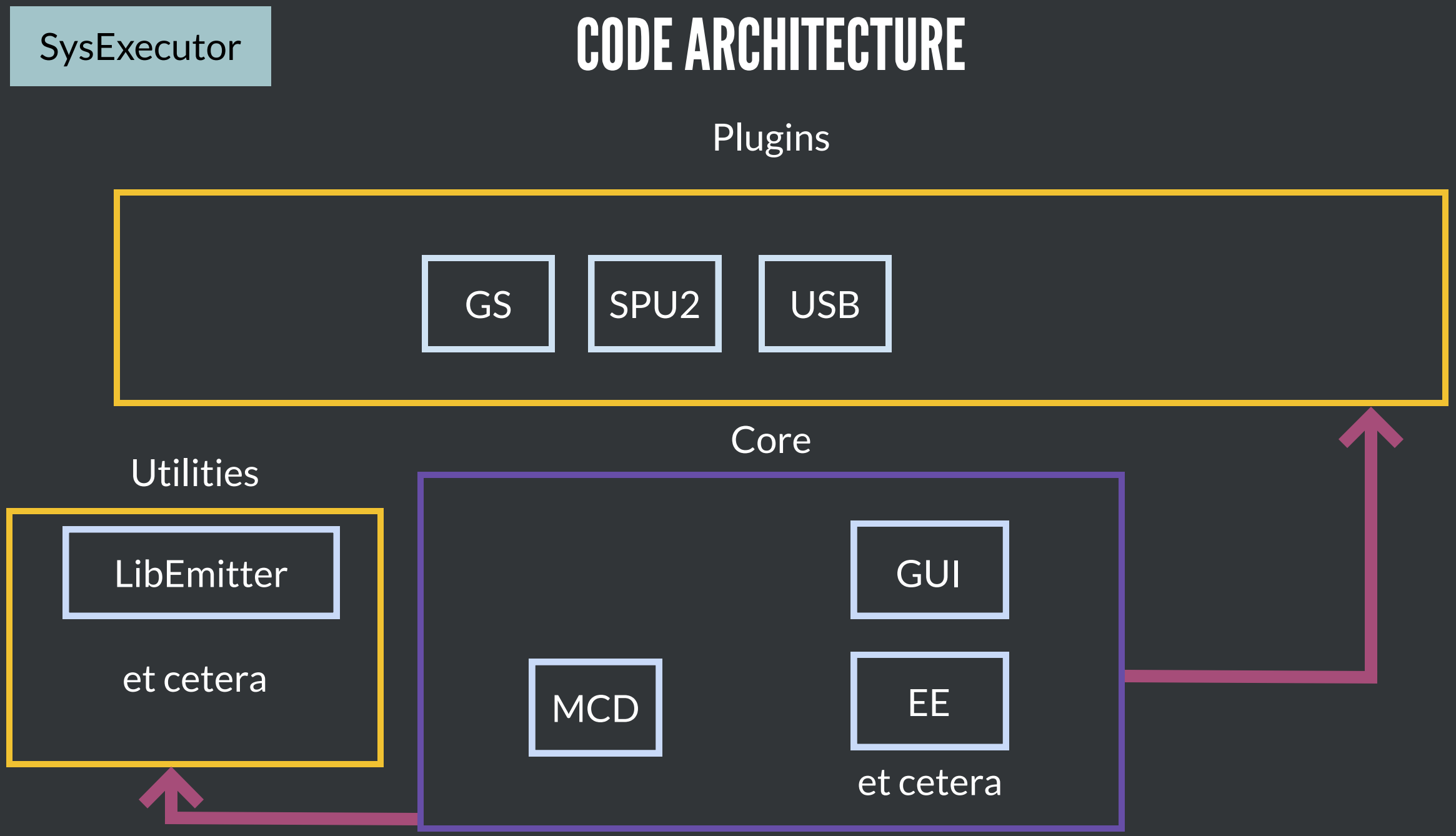
CODE ARCHITECTURE



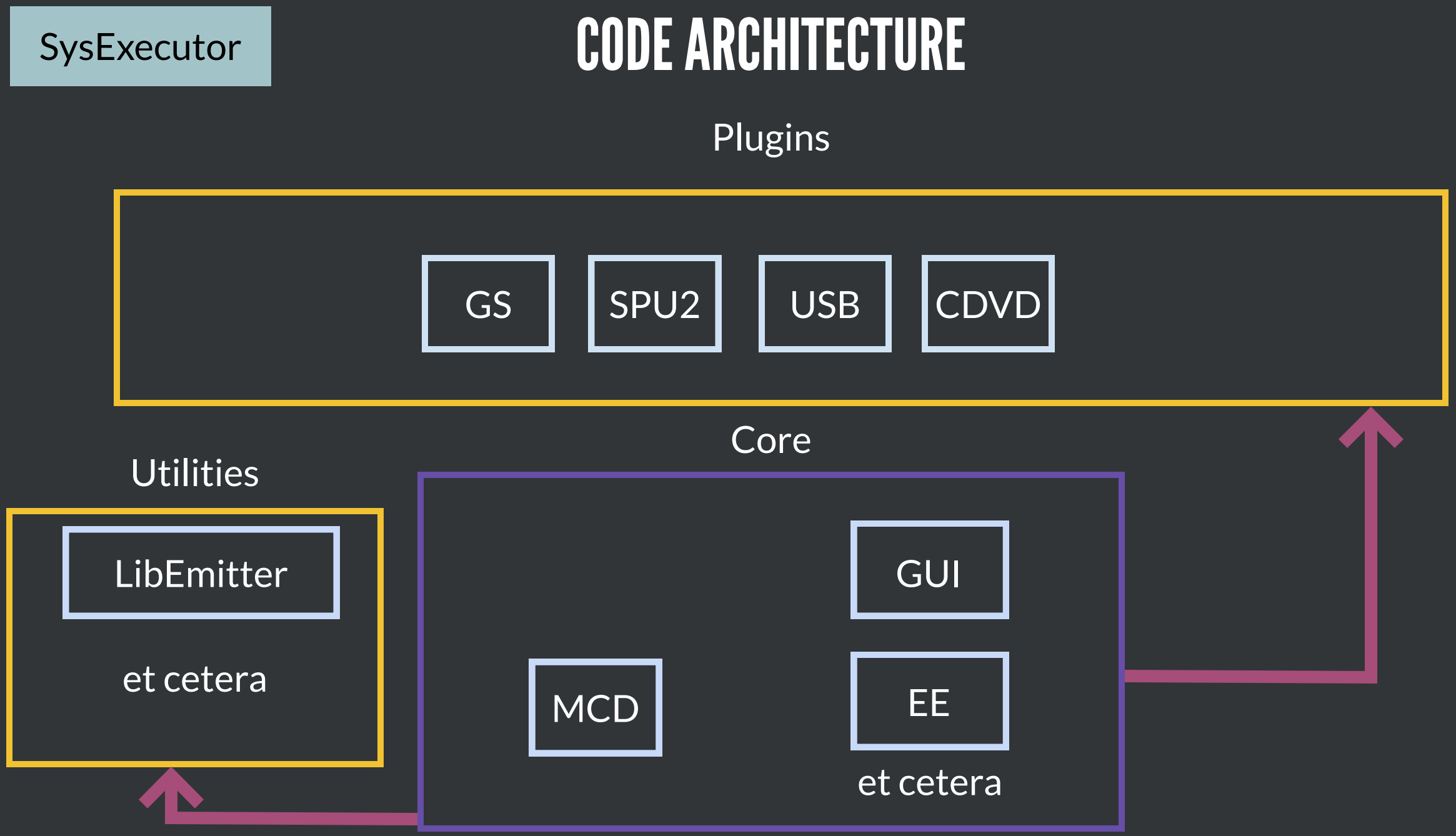
CODE ARCHITECTURE



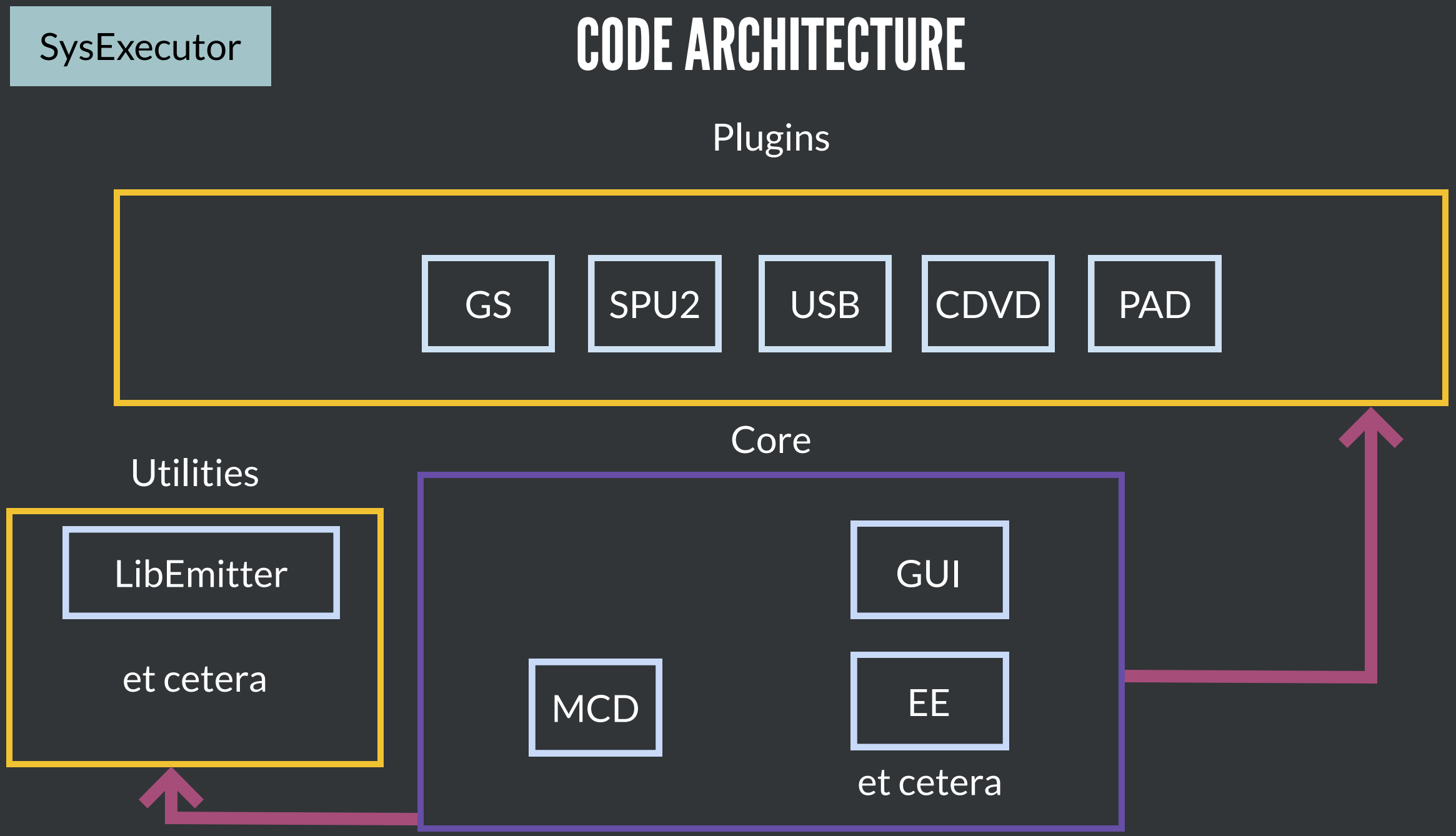
CODE ARCHITECTURE



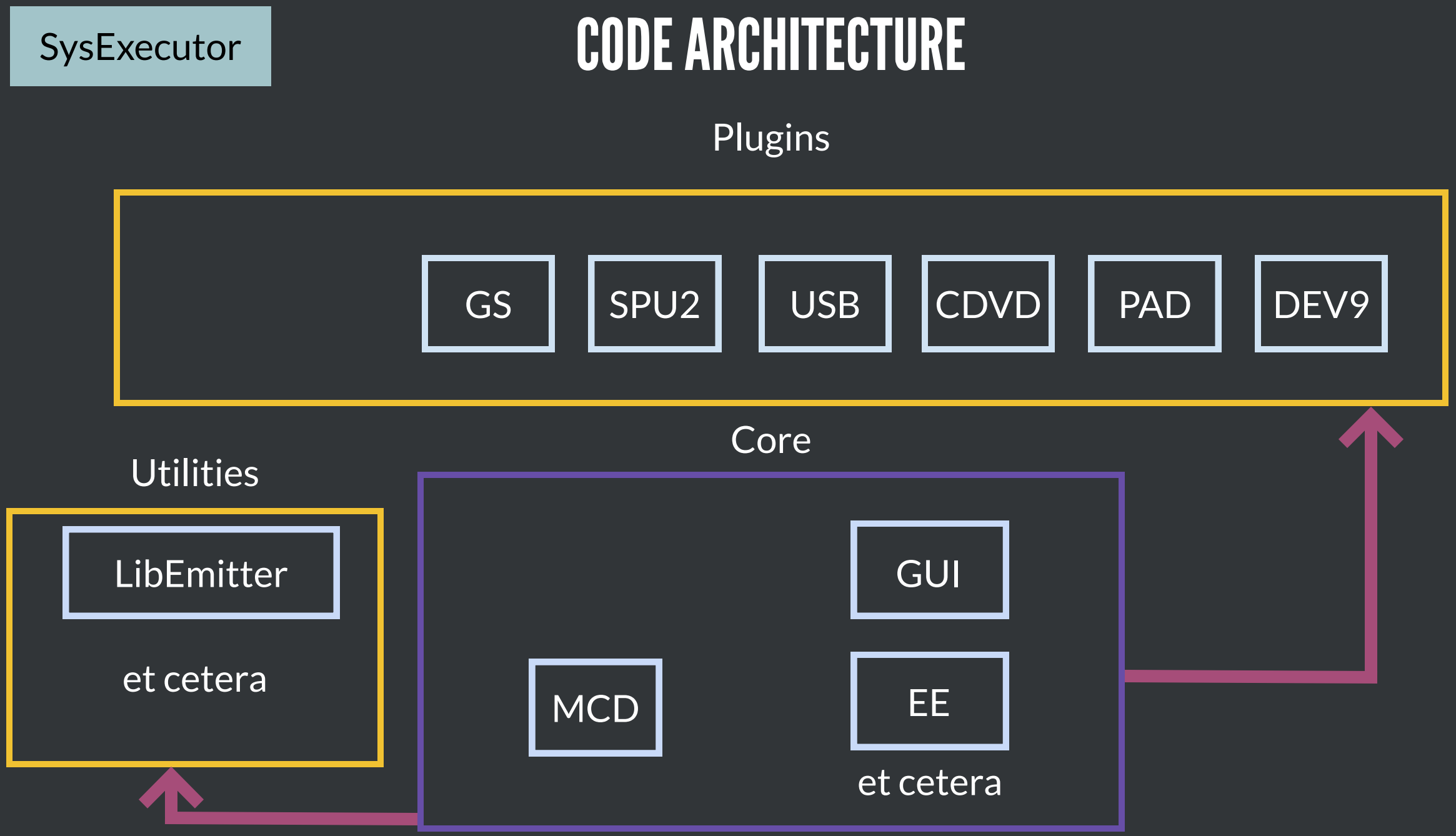
CODE ARCHITECTURE



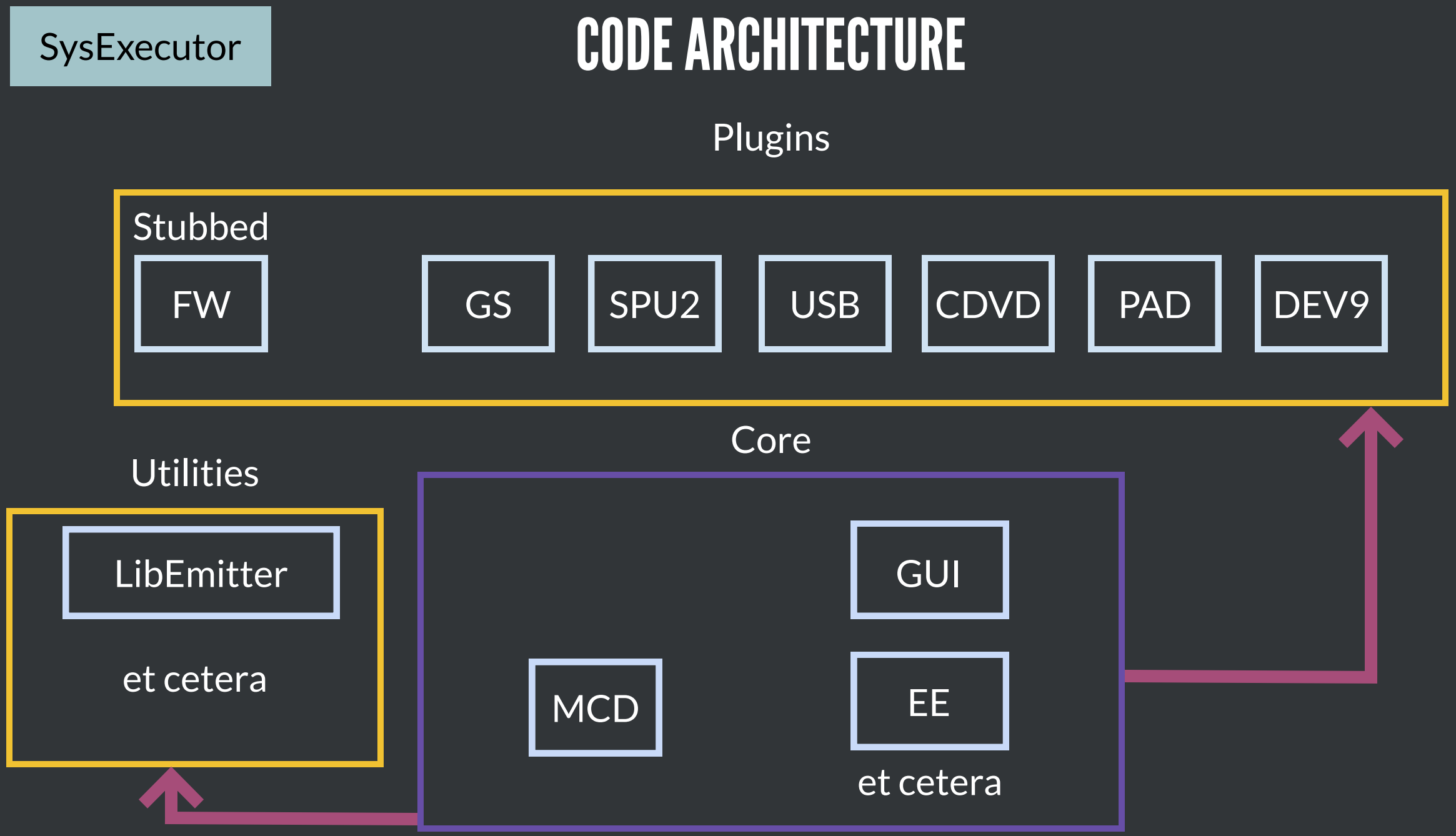
CODE ARCHITECTURE



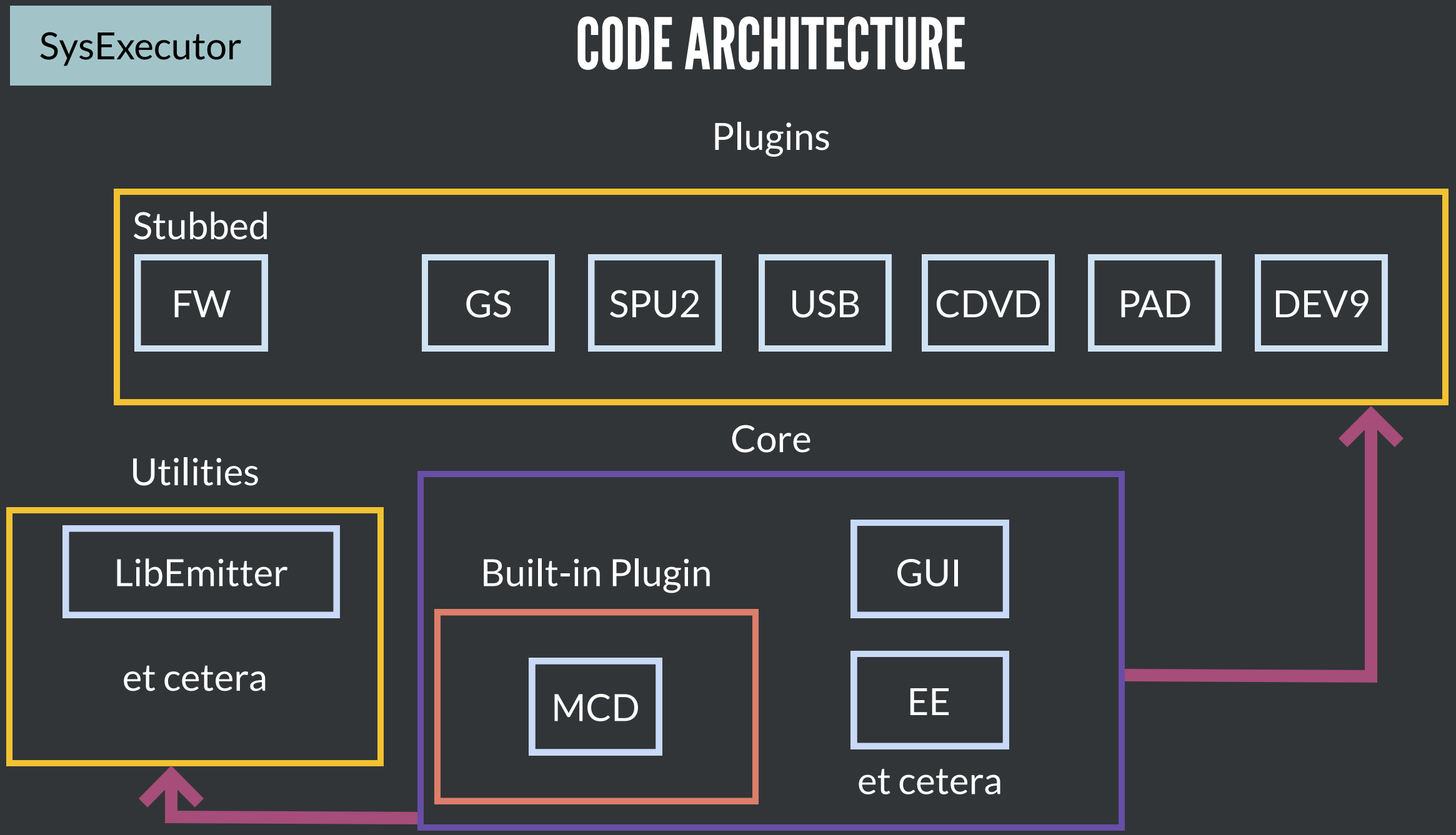
CODE ARCHITECTURE



CODE ARCHITECTURE



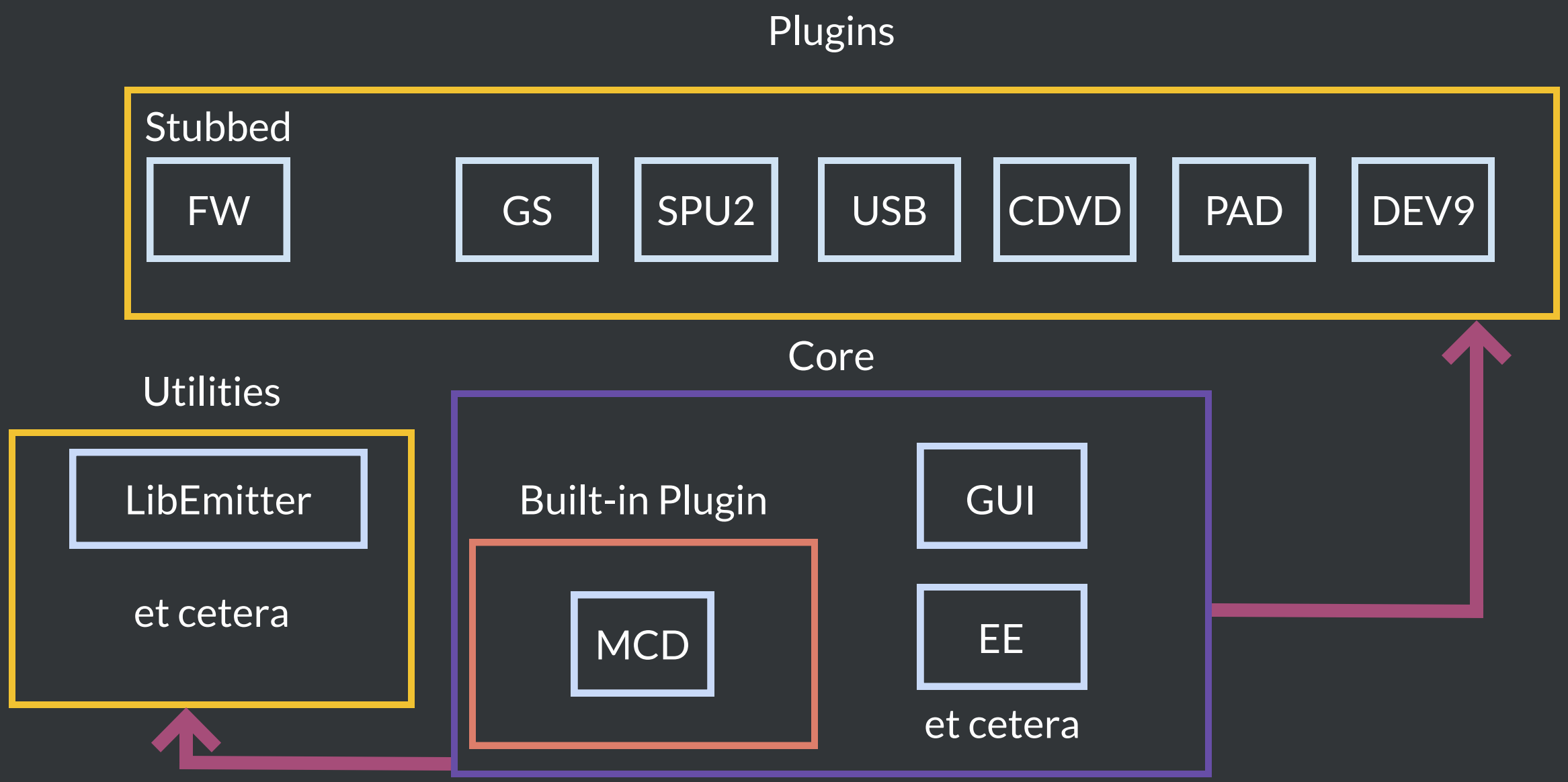
CODE ARCHITECTURE



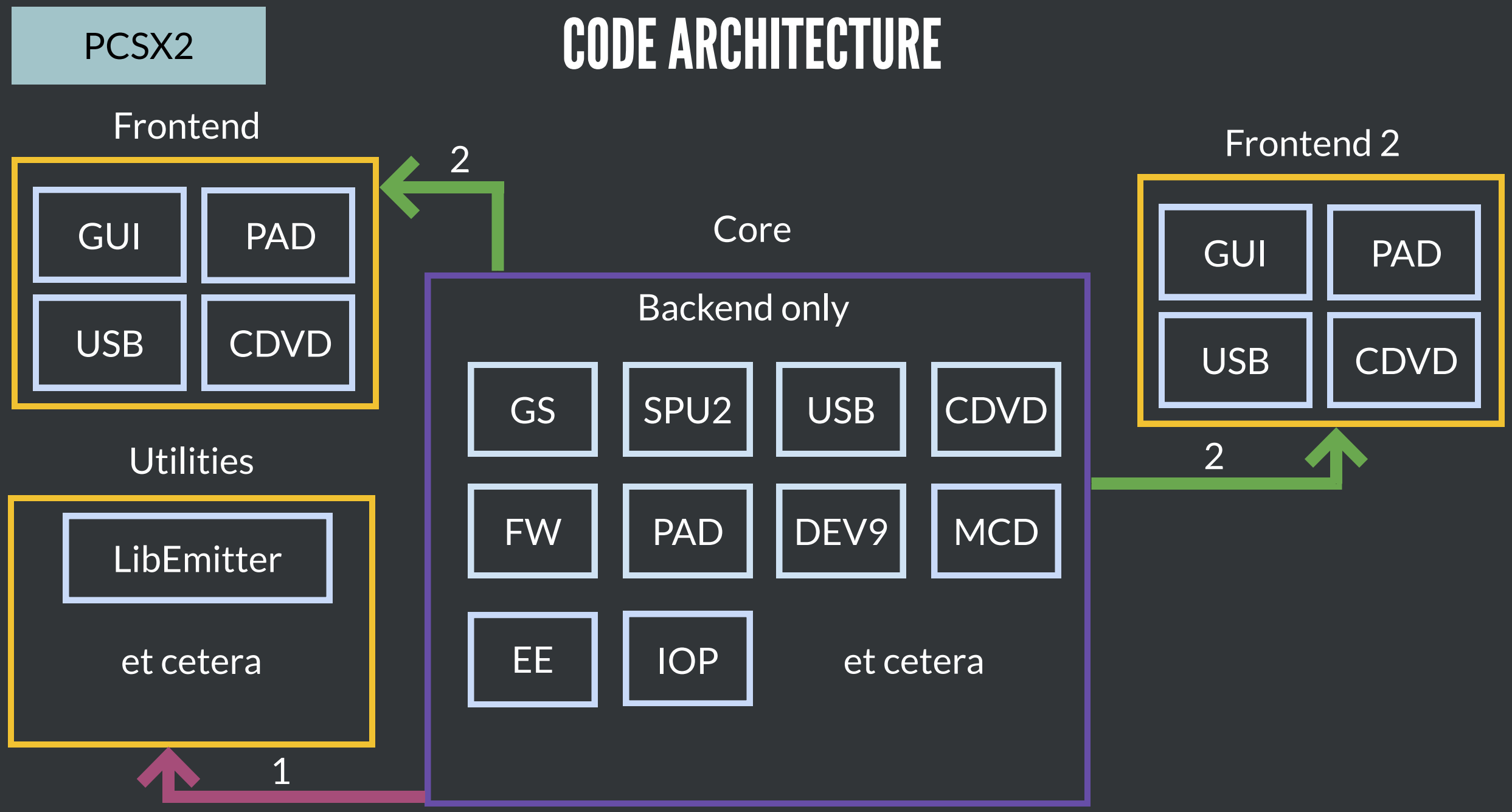
SysExecutor

CODE ARCHITECTURE

Hard dependency on wxWidgets!!



CODE ARCHITECTURE



JIT

JIT

With all the mentioned challenges, it will take a couple of months to get things working reasonably stable. By that time, more people would have switched to 64bit OSs. If we're even half right in our estimates, Pcsx2 will run much faster on a 64bit OS than on a 32bit OS on the same computer once x86-64 recompilation is done.

JIT

With all the mentioned challenges, it will take a couple of months to get things working reasonably stable. By that time, more people would have switched to 64bit OSs. If we're even half right in our estimates, Pcsx2 will run much faster on a 64bit OS than on a 32bit OS on the same computer once x86-64 recompilation is done.

PCSX2 64bit Recompilation

Created: 29 October 2006 Written by ZeroFrog

JIT

With all the mentioned challenges, it will take a couple of months to get things working reasonably stable. By that time, more people would have switched to 64bit OSs. If we're even half right in our estimates, Pcsx2 will run much faster on a 64bit OS than on a 32bit OS on the same computer once x86-64 recompilation is done.

PCSX2 64bit Recompilation

Created: 29 October 2006 Written by ZeroFrog

Oops

JIT

JIT

Fortunately, our 64 bit JIT is mostly done!

JIT

Fortunately, our 64 bit JIT is mostly done!

x64 Work and Testing #4102

tadanokojin 7d ago · 4 comments

tadanokojin 4d ago Maintainer Author

The memory card stuff should be sorted now, so Linux at least is ready to go. For GSdx, I understand tellow has it mostly working and **@GovanifY** is planning to take a closer look. Will need to sort out mipmapping but after those two we shall be ready for public release (and any fun bugs that come).

🎉 1

0 replies

Write a reply

JIT

Fortunately, our 64 bit JIT is mostly done!

x64 Work and Testing #4102

tadanokojin 7d ago · 4 comments

tadanokojin 4d ago Maintainer Author

The memory card stuff should be sorted now, so Linux at least is ready to go. For GSdx, I understand tellow has it mostly working and **@GovanifY** is planning to take a closer look. Will need to sort out mipmapping but after those two we shall be ready for public release (and any fun bugs that come).

🎉 1

0 replies

Write a reply

Unfortunately I'm giving a talk instead of fixing it :)

IPC

IPC

I've worked on a new protocol for
3 way game<->emulator<-> OS communication

IPC

I've worked on a new protocol for
3 way game<->emulator<-> OS communication

Merged

Socket IPC implementation #3591

tellowkrinkle merged 11 commits into `PCSX2:master` from `GovanifY:socket_ipc` on Sep 19, 2020

A 3 way communication can thus be established, game->OS; OS->game; OS->emu and game->emu.
game->OS IPC is poll based, OS->game event, game->emu poll and OS->emu event.
This is due to how the IPC is implemented: Dolphin IPC implements an event based IPC for emu<->game at the cost of having to modify the executable code of the game to implement this.
The upside of this PR is thus that you do not need to modify the game executable, only to reverse engineer it to find state variables and read off from it. As such this is not a no-cost implementation if the logic requires this.

IPC

I've worked on a new protocol for
3 way game<->emulator<-> OS communication

Merged

Socket IPC implementation #3591

tellowkrinkle merged 11 commits into `PCSX2:master` from `GovanifY:socket_ipc` on Sep 19, 2020

A 3 way communication can thus be established, game->OS; OS->game; OS->emu and game->emu.
game->OS IPC is poll based, OS->game event, game->emu poll and OS->emu event.
This is due to how the IPC is implemented: Dolphin IPC implements an event based IPC for emu<->game at the cost of having to modify the executable code of the game to implement this.
The upside of this PR is thus that you do not need to modify the game executable, only to reverse engineer it to find state variables and read off from it. As such this is not a no-cost implementation if the logic requires this.

Romhacks and game modding tools are about to get
a lot more interesting!

STATE

STATE OF THE PROJECT

STATE

STATE OF THE PROJECT

What to expect for 1.8:

STATE

STATE OF THE PROJECT

What to expect for 1.8:

- No Plugins!

STATE

STATE OF THE PROJECT

What to expect for 1.8:

- No Plugins!
- 64-bit Support!

STATE

STATE OF THE PROJECT

What to expect for 1.8:

- No Plugins!
- 64-bit Support!
- Reduced Input Lag!

STATE

STATE OF THE PROJECT

What to expect for 1.8:

- No Plugins!
- 64-bit Support!
- Reduced Input Lag!
- A new shiny IPC protocol!

STATE

STATE OF THE PROJECT

What to expect for 1.8:

- No Plugins!
- 64-bit Support!
- Reduced Input Lag!
- A new shiny IPC protocol!
- ...and much more (read our progress reports!)

STATE

STATE OF THE PROJECT

STATE

STATE OF THE PROJECT

What might be ready for 2.0:

- A New Qt based GUI along with support for pluggable & community GUIs

STATE

STATE OF THE PROJECT

What might be ready for 2.0:

- A New Qt based GUI along with support for pluggable & community GUIs
- Rework of our Infrastructure/Website

STATE

STATE OF THE PROJECT

What might be ready for 2.0:

- A New Qt based GUI along with support for pluggable & community GUIs
- Rework of our Infrastructure/Website
- Work on a pluggable JIT backend

STATE

STATE OF THE PROJECT

What might be ready for 2.0:

- A New Qt based GUI along with support for pluggable & community GUIs
- Rework of our Infrastructure/Website
- Work on a pluggable JIT backend
- A full cleanup of the codebase!

STATE

STATE OF THE PROJECT

What might be ready for 2.0:

- A New Qt based GUI along with support for pluggable & community GUIs
- Rework of our Infrastructure/Website
- Work on a pluggable JIT backend
- A full cleanup of the codebase!
- And hopefully other nice surprises ;)

END

CLOSING NOTES

END

CLOSING NOTES

We do not care about emulation wars

END

CLOSING NOTES

We do not care about emulation wars

It's always a tradeoff, we chose playability over accuracy (we still aim for accuracy)

END

CLOSING NOTES

We do not care about emulation wars

It's always a tradeoff, we chose playability over accuracy (we still aim for accuracy)

We all have different problems and different solutions

END

CLOSING NOTES

We do not care about emulation wars

It's always a tradeoff, we chose playability over accuracy (we still aim for accuracy)

We all have different problems and different solutions

Come hang out with us, chill and have fun, that's what emulation is all about!

END

CLOSING NOTES

We do not care about emulation wars

It's always a tradeoff, we chose playability over accuracy (we still aim for accuracy)

We all have different problems and different solutions

Come hang out with us, chill and have fun, that's what emulation is all about!

If you don't have fun, why even work on a project that you know you won't ever be paid for?

konami-code

THANKS

konami-code

THANKS

PCSX2 Team:

- refraction
- kotjin
- TellowKrinkle
- LightningTerror
- arcum42
- bositman
- jackun
- And others including past members like air and cottonvibes!

konami-code

THANKS

Friends:

- sirocyl
- ellie

PCSX2 Team:

- refraction
- kotjin
- TellowKrinkle
- LightningTerror
- arcum42
- bositman
- jackun
- And others including past members like air and cottonvibes!

konami-code

THANKS

Friends:

- sirocyl
- ellie

The PCSX2 Community:

- CK1
- Vaser
- RedDevilus
- pandubz

PCSX2 Team:

- refraction
- kotjin
- TellowKrinkle
- LightningTerror
- arcum42
- bositman
- jackun
- And others including past members like air and cottonvibes!

konami-code

THANKS

Friends:

- sirocyl
- ellie

The PCSX2 Community:

- CK1
- Vaser
- RedDevilus
- pandubz

Our Users

PCSX2 Team:

- refraction
- kotjin
- TellowKrinkle
- LightningTerror
- arcum42
- bositman
- jackun
- And others including past members like air and cottonvibes!

konami-code

THANKS

Friends:

- sirocyl
- ellie

The PCSX2 Community:

- CK1
- Vaser
- RedDevilus
- pandubz

Our Users

...And everyone else I forgot!

PCSX2 Team:

- refraction
- kotjin
- TellowKrinkle
- LightningTerror
- arcum42
- bositman
- jackun
- And others including past members like air and cottonvibes!

konami-code

THANK YOU!

@GovanifY

govanify.com

gauvain@govanify.com

@PCSX2

pcsx2.net

info@pcsx2.net