



Understanding the Bull GAMMA 3 first generation computer through emulation

PONSARD Christophe NAM-IP Computer Museum CETIC Researcher, DW Champion

FOSDEM 23 – Emulation – February 5

Context – NAM-IP Computer Museum

- Located in Namur/Belgium 30' from Brussels
 - worth a visit if you are staying a few days I Belgium after FOSDEM)
 - also: Pixel Museum (in BXL) & HomeComputerMuseum (in Eindhoven/NL not so far)
- Missions:
 - Preservation: safeguarding digital heritage, focus on local pioneers
 - Acquisition of artefacts, enriching collections: Bull, Burroughs/Unysis, I&B,...
 - Exhibitions: for all, specific animation, permanent/temporary
 - Research: about machines, software, communities
 here BULL GAMMA 3
- "Container design", an historical parallel









Outlook

- Historical context : a long time ago...
- Discovering the machine
- A look at existing emulators
- Our JAVA emulator
- Some lessons learned

Back to Early Electronic Machines with transition from electromechanical \rightarrow electronic machines



48: Bull BS120

49: IBM 407

Inside the machine: sources ?

• Only a few left (Anger, Grenoble, Frankfurt, some pieces in Namur)

Liste o

• Documentation: ACONIS/Bull Belgium





• Emulators !

DOSBox	0.74	-3, Cpu	spee	ed: ma	ax 1	00% d	ycles,	Frame	skip	0, Pro	ogram	n:	GE				-		×
MD=0 MS1=0 MO=0 0	Gi U O O	amma Jersi 0 0	Emu on 0	laTi 0.81 0 0	on .ß 0_0	0 0	Sér Sér Sér	ien ien ien	•3, •3, •3,	NL: NL: NL:	00→ 01→ 02→	43 43 43	B (A ? 9 4): KI): KI 1: KI	30 39 34	⇒3,1 ⇒3,1 ⇒3,9	1 0		
M2=0 0 M3=0 9	00	00		0 0	0	00	Sér Sér	ie n ie n	•3, •3,	NL:	04→ 05→	13 13	71	7 : KI	3 7	⇒3,0 ⇒3,7 ⇒3 6			
M4=0 0	0 0	0 0	0	0 0	0	0 0	Sér	ie n	•3,	NL :	06+	43	5 :	L : KI	3 1	.⇒3,5			
M5=0 0	0 0	00		0 0	0	00) Sér			NL:	07≁	43	4 (9 : KI	30	⇒3,4			
M6=0 0	0 0	0 0	0	0 0	0	0 0	Sér	ie n		NL :	08 +	43	3 9	9 : KI	3 9	₩3,3			
M7=0 0	0 0	00	0	0 0	0	00) Sér	ie n	°3,	NL :	09+	43	Z 3	3 : KI	3 3	₩3,2			
DNT 1-00						- 94	inia		(ть	blea					mel	_			
RNL2=00	NL	TADF	4	•437	7	4323	3 120	1 23	41	1201	23	41	120:	1 23	341	1201	2341	1201	234
		43B0		•436	4	4315	5 234	1 12	01	2341	12	91 3	234:	1 12	201	2341	1201	2341	120
NL=C6 N	L:1	4 3A9		•435		4301	120	1 23	41	1201	23	11	120	1 23	341	1201	2341	1201	234
N	L:2	4394	- 7	→4 34	0	1D01	234	1 12	01	2341	12	91 ;	234:	l 12	201	2341	1201	2341	120
• • N	L:3	•4382		•433	19	2341	120	1_23	41	1201	23	41	120:	1 23	341	1201	2341	1201	000
		No	tat	ions		A=r	1=10,	B=A	=11	, C=	P=1.	۷,)=X=	=13,	, E=	R=14	, r=z	=15 —	
Det-8	330	a 630	0 1	ambc FAA	12	- 002	001Za 74 67	100 56 C	n-0 767	, ri	ste	n- 511:	2 A	5100 713	: II- (733	4 12	.0000	0000	666
ade 9	6C01	0000 0000	68	400	13	100	1 67	78 7	005	ade	. q.	BA4	2 Af	131	AA1	1 13	00000	0000	000
n° 10:	6F0	003	28	500	14	:8AG	01 67	45 8	A8A	n°	10:	100) 20	000	1D0	0 14	0000	0000	000
0 11:	600	à 508	38	700		:A01	LA C7	78 6	052		11:	000	0 00	000	000	0 15	0000	0000	000
					ÉC	HAP-	Quit	ter		F4	Exé	cut	er	iuso	m'à		F7-P	as-à-i	nas

instructions	Supervision	Controler l'execution du pr	rogramme
~	Aide		
		Execution	< Série 3
x Boox 9xxx Axox Bxxx x Dxxx Exxx Pxxx	Programme "Raci Sortie 1 : 0000 Programme "Suit Sortie 0 : 0000 Sortie 1 : 0000	ne Carree" chargé 10000000 e De Fibenacci" chargé 10000001 10000001	0 : 1a00 c0 32 1 : 1c07 c0 33 2 : 3200 z8 34 3 : 3300 z8 34 4 : 3100 z8 36
angement de Serie - Saute à la ligne 0 ent la ligne de laquelle on saute + 1 en	TH	Rer Continuer	5 : a001 AN 37 6 : 8400 BO 38 7 : 6300 BO 38 9 : 8500 AN 40
"hancement de Sarie Saute à la liene 0		Controle	10 : 0400 80 42 : 11 : 8100 08 43 : 12 : 6500 80 44 : 13 : 8400 08 45 :
retient la ligne de laquelle on saute + 1 en	NL CII NS1 0 ND 0	M0/M1 0000000000000000000000000000000000	14 : 8e00 08 46 15 : 6200 80 47 16 : a001 AN 48 17 : 420 08 49
Changement de Serie - Saute à la ligne 0	RNL1 00 RNL2 00 Octade 7	N5 000000000000000000000000000000000000	18 : 8400 08 50 19 : 1800 E51 51 20 : 1801 E51 52
Retour Serie VRS - Variante Retour Serie gne enregistrée en RNL1	Seizaine O	Node Decimal	21 : 900f CN 53 22 : 011d V 54 23 : 0000 V 55 24 : 0000 V 55
letour Serie VRS - Variante Retour Serie ne enregistrée en RNL2			26 : 0000 V 58 27 : 0000 V 58 27 : 0000 V 59 28 : 0000 V 60
nte Retour Serie - Instruction non	м8	Octade 0	29 : 0000 v 61 30 : 0000 v 62 31 : 0000 v 63

Hardware (initial version)

First generation computer:

• vacum tubes, delay lines

Code stored in a connection panel (64 instructions)

- So not Von Neumann (pgm need to be in memory)
- although somehow "memory mapped" on "serie 3"

Main memory: 7 registers

- Delay lines
- Vacuum tube for regeneration
- 8.5 kg to store... 6 bytes

ALU:

- vacuum tubes (low reliability) → germanium diodes
- + → * % performed internally by iterative + -

Clock: 2.5 Hz... because synchronised with motor of motor of the connected electromechanical equipment !

ADDITIONAL

Nice "drawer" design easing maintenance !



GAMMA 3 : Computer and/or Calculator ?

• Initially: designed as calculator « slave » for tabulating machine

"In its current configuration, **the computer acts as an extension unit for the punched card machine it is connected to**. The cards are read in the reader station which transmits data to the computer. The computer in turn performs all necessary calculations and transmits the results back to the punched card machine which will print or punch these values. Regardless of the task, **the computer is so fast that there is no visible delay** caused by the calculations" (Pierre Letort, 1953)

So programs = specific calculation routines called by tabulating machine

- Evolution:
 - 3M: scientific version with floating point support (otherwise need subroutines)
 - ET (1957): drum extension (about 100Kb) → program NOW truly in memory
- Finally: GAMMA 3 : first French computer

→ GAMMA 3 becomes the central unit

and tabulating machine becomes the peripheral device (= transition !)



Should a Computer Operate in Decimal or Binary ?

- AVANTAGES ET INCONVENIENTS DU BINAIRE.

1°) Avantages

Ne comportor que deux chiffres : 0 et 1 ost un argument puissant en faveur de l'emploi du système binaire dans les calculatours. En effet, il est très simple de représenter alors ces 2 chiffres sur une machine. Par exemple :

- on peut convenir qu'un relais au travail signific 1, et au repos signifie 0.
- on peut égalemont dire qu'une lampe allumée signifie 1, ct éteinte 0.

Sur le Gamma, le 1 sera symbolisé par la présence d'une impulsion électrique, le zéro par son absence.

Le binaire possède un autre avantage très appréciable sur les machinos : la simplicité des opérations arithmétiques. Ainsi, la table d'addition se réduit à 3 égalités :

0 + 0 = 0 0 + 1 = 1 1 + 1 = 0 + roport sur le rang. suivant.

(Au lieu de ces 3 égalités, la table d'addition en décimal en comporte 100, de :

0 + 0 = 0à 9 + 9 = 8 + report).

On comprond que la réalisation des calculs dans une machine en soit très simplifiée.

Cet avantage est encore plus flagrant dans le cas de la multiplication dont la table s'écrit :

0 x 0 = 0 0 x 1 = 0 1 x 1 = 1

2°) Inconvénients.

La vie courante utilise le système décimal et il est bien évident qu'il n'est pas question de changer cet état de choses. D'ailleurs, un nombre binaire est beaucoup plus long que son équivalent décimal et ne permet pas de se rendre compte à première vue de l'ordre de grandeur qu'il représente. Ainsi 1025 a pour équivalont binaire un nombre de onze chiffres, pou pratique à interprétor.

Les données de départ d'un calcul seront donc presque toujours écrites en décimal et les résultats, pour être expleitables, devront également être restitués en décimal. On ne conçoit pas une machine comptable qui établirait des feuilles de paye écrites en binaire.

Tout calculateur utilisant le binaire dovra donc comporter :

a) un organe effectuant la traduction des données de décimal en binaire.

b) un organe faisant l'opération inverse sur les résultats.

Pro:

- Only two figure: powerful
- Maps easily to relays !

Cons:

- Longuer
- Needs to translate back & forth with decimal (for humans

Conclusion ?

- Use « semi-decimal » → binary-coded decimal !
- Char = 4 bit to code 1 BCD (or 1 HEX)

Note: later extension (ET) supported full binary mode

GAMMA 3 – Core "banale" memory : a few registers

- M0=M1: "accumulator" IN/OUT
- M2..M7: only IN
- M8..M15: "extra memory" (switched "octade" see extension)
- Technology: delay line
 - 1 register = 1 word
 - 1 word of 12 chars of 4 bits (BCD or HEX)

→1 word = 6 bytes (note: longer than 32bit integer)

→M1..M7 = 42 bytes of "main" memory



GAMMA 3 Complete Architecture

Note also:

PC, CMP, shift...

Used in instructions



GAMMA 3 – Structure of Instruction Set

Structure: 4 chars

- TO: Type of Operation
 - E.g. AN (add), SN (subtract), MR (reduced mult.), MC (complete mult.),...
- AD: <u>ADd</u>ress
 - Memory register involved in operation
- OD: Ordre Début (i.e. start position)
- OF: <u>Ordre Fin</u> (i.e. end position)
 - Because operation can consider a subrange of a Word !

1 Word = 12 char → so possible to store 3 instructions/word



Coding sheets

			E	3 L	js	CO	mpris	en	Proble	ēn 1	^{re:} B et 1, 9		3.10 ³ V3	Pl. 2
Sél	ection	NL	TO	Co	des	5				_				
	gnes	10	10	AD	OD	OF	Operation	MD			M 2	MB	Ma	
++	++++	Ca	rte		[°)				10864	120	01086420	0108642	01086/121	
++	++++	-	-	-	-			0	++++++++	+++		VILLEN	Lin ABind	
++	+ + + + +	-	4	2	7	1	KB		+++++++++++++++++++++++++++++++++++++++	+++-	+++++++++++++++++++++++++++++++++++++++			
++	++++	9	4	2	6	7	KP	0	+++++++++++++++++++++++++++++++++++++++	++++				
+	++++		4	2	5	3	KO	0	++++++++	+++		HILLUI I		
			4	2	4	2	KD	0	+++++++++	++++				
							AB_	0	++++++++	++++	111111			
1 1			6	3		1.		-	+++++++++	++++	+++++++++++++++++++++++++++++++++++++++			
		8				4	BO	0	+++++++++	+Y	+++++++++++++++++++++++++++++++++++++++			1
1 1								-	+++++++	++++				1
++	+++		10	0				-	+++++++++					1
+ +	+++		12	2	4	8	MR	0	HIIIH YI	V3				7
++	+++								++++++++	++++				7
++	+++		6	_1	2		BO	2	111144	E.			· · · · · · · · · · · · · · · · · · ·	7
++	+++	7	8	2	4	10	OB	4		++++	Y. V3	hin hin		+
+ +	++++		6	4	3	6	BO	3	U	B				+
++	111		9	2	6	9	CN	6	UB.K	23	Multipli	cation par 1	03	+
										 		+++++++++++++++++++++++++++++++++++++++	<u>+++++++++++++</u>	+
1 1									1.1.1.1.1	++++	1	+++++++++++++++++++++++++++++++++++++++	+++++++++++++++++++++++++++++++++++++++	++
		6								++++	****	+++++++++++++++++++++++++++++++++++++++	+++++++++++++++++++++++++++++++++++++++	+
+	++++	0						1	****	++++	+++++++++++++++++++++++++++++++++++++++	* * * * * * * * * * * * * * * * * * * *	+++++++++++++++++++++++++++++++++++++++	++
++	++++							-	****	++++	+++++++++++	+++++++++++++++++++++++++++++++++++++++	+++++++++++++++++++++++++++++++++++++++	++
+	++++		13	2	4	10	DR	0	+++++++	++++	+++++++++++++++++++++++++++++++++++++++	+++++++++++++++++++++++++++++++++++++++	+++++++++++++++++++++++++++++++++++++++	+++
+	++++					-		-	+++++++	++++	+++++++++++++++++++++++++++++++++++++++	+++++++++	+++++++++++++++++++++++++++++++++++++++	+++
'			-			2	00	0		B	decimates,	L. L. L. L.	1 1 1 1 1 1	1

Exploring the Instruction Set (excerpt)

AXXX – addition (AN) 4XXX – constant transfer to memory OXXX – V variant • AD: mem 0000 NOP ;-) OD: position • 01xx JUMP if > flag set BXXX – subtraction (SN) OF: value • 02xx JUMP if = flag set • O3xx JUMP if >= flag set CXXX – reduced multiplication (RM) 5XX – copy between octads • 04xx JUMP if neg flag set 6XX – transfer to operative mem. (BO) DXXX – reduced division (RD) 1XXX • AD: mem • OD & OF: range • 11/12/13xx: VCS: serie switch EXXX – full multiplication (MC) • 15/16/17xx: VRS: serie return • 18/19xx: out to connected machine ^{7XX} • 71/72xx: shift memory FXXX – full division (FD) • 1Axx: decimal mode (CD) 73/74xx: logical AND • 1Fxx: binary mode (CB) (what about OR ?) • ... 8XX – transfer from operative mem. (OB) Note: • AD: mem 2XXX – drum transfer (BT) "reduced" * / for "small numbers" • OD & OF: range \rightarrow range need to be manager ... • "complete" * / operation using 9XXX – comparison operations 3XXX – setting memory to zero M1+M2 as extended storage ...

A look at the code card

- "ordonnateur"
 "ordinateur" in French was coined I year later by J. Perret for IBM "god puts world in order"
- Not so easy to decode...

										2													M	\sim	Р	\times	R	2		
								Série	OF	Α [) —	-	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	-	AD
								0	0	4	8	12	Jumais	>	=	>	Ms				9,14	15,14	1,14	0,14	11,14	12,14	13,14	14,14		
								1	1	5	9	13	Toujours	\$	≠	<	Ms				8,3	9,3	0,3	11,3	12,3	13, 3	14,3	15,3		
								2	2	6	10	14	Vo	V1	V ₂	V3	V4	V5	V6	V7		Vac	Vac	Vac	Vac	Vac	Vac	Vac	U	V
	TT		- T	0 =	= 2			3	3	7	11	15	ν _o	V1	Īv2	Īv3	V4	V5	V6	V7		Vac	Vac	Vac	Vac	Vac	Vac	Vac		
	0	0	0	0		0	0	Ω	0	1	2	3	VCS	VCS	VCS	VCS		VRS	VRS	VRS	ES4	ES2		2	00	CS ₂	5	CB	1	
	1	2	1	1		1	2	1	4	5	6	7	AMD	RAL	BD			RAL	HALS				IL		IL	(OF)			7	
	2	2	2	2		2	4	2	8	q	10	11	(01)		100	(1111)							UF	//////	M 2				~	_
	2	4	2	2		Z	6	2	12	13	11	15		0)		1		1			2	32	15		_	٨	D		
CS-	5	0	5	5		5	0	5	12	15	14	15		1111					1			2	50	115			A	U	7	70
0	.4	8	4	4	BT	4	8	4	16	17	18	19					Rā	Z			Rà	Z	-		Ма	intie	en N	D	3	ZB
	5	10	5	5		5	10	5	20	21	22	23	Em	ISSIO	00-8	BV.	Filtr	ee [- M1	Filtr	ee [MB				m	4	KB
	6	12	6	6		6	12	6	24	25	26	27	OF		Rece	pteur													5	GG
	7	14	7	7		7	14	7	28	29	30	31	0	F →	M	1	Ма	intie	en' f	. M1	M	B -	N	11				D	6	BO
123	0	1	8	8		0	1	8	32	33	34	35					Rd	ΞZ	F.	M1	R		F. MF	МВ	0	pérc	tion	15	8	OB
Ш	1	3	9	9		1	3	9	36	37	38	39	Supp	res N	1₁ ≷	OF	M1	t. ?	2	M1 F.	M	11 2	ME	3		avi	ec		9	CN
MU	2	5	10	10		2	5	10	40	41	42	43	cadra preak	ige ible M	11 +	OF		M1 >	< 2		M	1 +	ME	3	P	réal	able	2	10	AN
WO	3	7	11	11		3	7	11	44	45	46	47	Maint	M	1 -	OF	R	àΖ	f.	M ₁	M	1 -	ME	3	1-		۹U =	=U U	11	SN
Ш	4	9	12	12	ТВ	4	9	12	48	49	50	51	M	×C	F =	M ₁	1111	(((()			M1	×M	B =	M1	No	mbr	e d	le	12	MR
AIN	5	11	13	13		5	11	13	52	53	54	55	M	i÷0	F =	M ₁					M1	÷M	B=	M ₁	de	M ₁	= 1	ÖD	13	DR
EIZ	6	13	14	14		6	13	14	56	57	58	59	Ma	xOF	= M	1 M2					illi		MexME	B = M1 Ms	NO	mbr	e d	le	14	MC
S	7	15	15	15		7	15	15	60	6.1	62	63	Mal	M2 ÷	OF	=M-							Mi Ma÷	MB=M2	de	ecal M ₁ -N	dge	s MD	15	DC
	Grou	AD	Piste	in		Bloc	0 F	d'n		N				()		(1111)		1	./////		2	30	15	-	-	A	D	T	0

E.T. ORDONNATEUR

TABLEAU DE CODE -

GEYNE

CODE 381

Emulator by V. Joguin (1990's)

- Written in x86 assembly code by Vincent Joguin (1995)
- Now need an emulator (thanks DOSBox !)

http://vincent.joguin.com/ GAMMAET.ZIP

🚟 DOSBox 0.74-3, Cpu speed: max 100% o	ycles, Frameskip 0, Progran	n: GE	– 🗆 X
MD=0 Gamma EmulaTion MS1=0 Version 0.81ß MD=00000000000000000	D Série n°3, NL:00→ Série n°3, NL:01→ Série n°3, NL:02→ Série n°3, NL:03→	4 3 B 0:KB 0→3,11 4 3 A 9:KB 9→3,10 4 3 9 4:KB 4→3,9 4 3 8 2:KB 2→3 8	
M2=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	Série n°3, NL:04→ Série n°3, NL:05→ Série n°3, NL:05→	4 3 7 7 KB 7→3,7 4 3 6 4 KB 4→3,6 4 3 5 1 KB 1→3,5	
M5=000000000000000000000000000000000000	Série n°3, NL:07→ Série n°3, NL:08→ Série n°3, NL:09→	4 3 4 0:KB 0→3,4 4 3 3 9:KB 9→3,3 4 3 2 3:KB 3→3,2	
RNL1=00 Se	rie n°3 (Tableau d 1201 2341 1201 23	e connexions) —	7341 1201 2341
A :0→43B0 5→4364 4315 NL=C6 NL:1→43A9 6→4351 4305	1201 2341 1201 23 2341 1201 2341 12 1201 2341 1201 23	01 2341 1201 2341 41 1201 2341 1201	1201 2341 1201 2341 1201 2341
NL:2→4394 7→4340 1D0: NL:3→4382→4339 2343	2341 1201 2341 12 1201 2341 1201 23	01 2341 1201 2341 41 1201 2341 1201	1201 2341 1201 2341 1201 0000
Tambour : S	eizaine n°0, Piste	n°0, Bloc n°0 —	F-2-15
Oct-8:3300 6300 1F00 12:C07 ade 9:6C00 1C06 8400 13:1C0	4 A756 C767 Oct-8: 1 A778 7005 ade 9:	A112 A713 C734 12: BA47 A031 A011 13:	0000 0000 0000 0000 0000 0000
n° 10:6F00 0032 8500 14:8A0 0 11:600A 5083 8700 15:A0	1 6745 8A8A n° 10: A C778 6052 1 11:	1000 2000 1D00 14 0000 0000 0000 15	0000 0000 0000 0000 0000 0000
ECHAP-	Quitter <mark>F4</mark> -Exé	cuter jusqu'à	F7-Pas-à-pas 🚽

BullGammator (ACONIT, 2020)

- Open Source : <u>https://github.com/lutrampal/bullgammator</u>
- Web-base written in Javascript (nodeJS), very well documented
- Include panel interface, library, console (with stepping mode "titiller")
- Nice code base for emulator core with unit tests
 analysed and mainly transposed in JAVA

https://www.aconit.org/histoire/Gamma-3/Simulateur/	▣ ☆
NE TO AD OD OF Distrib TO AD OD OF NE	

https://www.aconit.org/histoire/Gam	ma-3/Simulateur/					1
		Execution	<	Série 3	>	
	Programme "Equat Programme "Racin	ion 2nd Degre Deca" chargé e Carree" chargé er Continuer	 0 :: 1ct 1 :: 633 2 :: 8d6 3 :: 188 4 : 1at 5 :: 633 6 :: 900 7 :: 855 8 :: 033 10 :: a44 11 :: c00 12 :: 700 13 :: 855 	$\begin{array}{cccccccccccccccccccccccccccccccccccc$: 6200 BO : a009 AN : a700 AN : c010 MR : c0a5 MR : 610 BO : 5110 BO : 5120 BO : 9520 CN : 8520 OB : 0271 V : 6500 BO : 3200 ZB : 1c07 CO : 8800 OB	
	NL CO MS1 O MOD O RKL1 OO RKL2 OO Octade O Seizaine O	M0/M1 0000000000 M2 0000000000 M3 01440000052 M4 00000000000 M5 0000000000 M6 0000000000 M7 00000000000 Mode Décimal	14 : 901 15 : 633 16 : 044 17 : 033 18 : 700 19 : 814 20 : 864 21 : 033 223 : 040 23 : 040 24 : a00 25 : a14 26 : 26 26 : 72 26 : 72	b5 CN 46 20 R0 47 20 V 48 50 V 48 50 V 49 03 AMD 50 02 08 52 61 V 53 92 AN 54 69 V 55 0- -AN 57 00 AMD 58	: 1801 ES1 : 0000 V : 0000 V	
	M8 M9 M10 M11 M12 M13 M14 M15	Octade 0 00000000000 00000000000 00000000000 00000000000 00000000000 00000000000 00000000000 00000000000 00000000000 0000000000000 00000000000 00000000000 00000000000 00000000000 00000000000	28:66 29:32 30:70 31:77	00 B0 60 00 ZB 61 10 ZB 61 10 AMD 62 00 DC 63	: 0000 V : 0000 V : 0000 V : 0000 V : 0000 V	



3D Simulator (by Aconit)





A look at the addition – carry algo (dec/bin fine)

```
/**
add the given memory to this one - this works both in decimal and binary mode
(param other the memory that should be added
@param from index of the block from which the addition should start
Oparam to index of the block to which the addition should end (excluded)
@param overriding carry if true, at the end of the addition, the resulting carry out will override the next
memory block if it is not null. Otherwise it will be added to the next memory block.
   */
public void add (Memory other, byte from, byte to, boolean overriding carry) {
         assert (from >= 0) : "from should not be negative";
         assert (from < to): "from should be inferior to "+ to;
         assert (to <= this.blocks.length) : "to should be inferior to the number of blocks per memory";
         int carry = 0; // implementation simulates the real work across memories
         for (int i = from; (i < to) || ((carry == 1) & loverriding carry); i++) { // TODO check precedence
               int other val = i < to ? other.blocks[i] : 0;</pre>
               int res = this.blocks[i%this.blocks.length + (this.blocks.length - NB CHRS PER WORD)] + other val + carry;
               if (res >= this.getMode().base) {
                     carry = 1;
                     res -= this.getMode().base;
               } else {
                     carry = 0;
               this.blocks[i%this.blocks.length + (this.blocks.length - NB CHRS PER WORD)] = (byte)res;
         if (overriding carry && (carry!=0)) {
               this.blocks[to%this.blocks.length] += carry;
         }
```

A look at the subtraction \rightarrow Java substract !

```
/**
```

```
subtract the given memory to this one
Oparam other the memory that should be subtracted
Oparam from index of the block from which the subtraction should start
(param to index of the block to which the subtraction should end (excluded)
  */
public void subtract(Memory other, byte from, byte to, byte this from, byte this to) {
      assert (from >= 0) : "from should not be negative";
      assert (from < to) : "from should be inferior to to";
      assert (to <= this.blocks.length) : "to should be inferior to the number of blocks
      per memory";
      long valM1 = this.getDecimalValue(this from, this to) - other.getDecimalValue(from,
      to); // <u>implem</u> is done through translation to java long (int not enough !)
      this.setDecimalValue(Math.abs(valM1), this from, this to);
      if (valM1 < 0 && this.getMode() == MemoryMode.DECIMAL) {</pre>
          this.bullGamma.ms1 = 10;
```

A look at the division

```
/**
divide the given memory to this one
(param other the memory that should be divided
@param from index of the block from which the division should start
@param to index of the block to which the division should end (excluded)
   */
public void divide(Memory other, byte from, byte to) {
        long vmb = other.getDecimalValue(from, to);
        if (vmb == 0) {
              throw new Error("Division by 0.");
        while (this.bullGamma.md > 0) {
              while (this.getDecimalValue((byte)(from + this.blocks.length - NB CHRS PER WORD), (byte)this.blocks.length) <</pre>
              vmb && this.bullGamma.md > 0) {
                   this.shiftLeft();
                   this.bullGamma.md--;
              while (this.getDecimalValue((byte) (from + this.blocks.length - NB CHRS PER WORD), (byte) this.blocks.length)
              >= vmb) {
                   this.blocks[0]++;
                   this.subtract(other, from, to, (byte)(from + this.blocks.length - NB CHRS PER WORD),
                   (byte) this.blocks.length);
```

Current Experimentation Environment

IDE (Eclipse) Fibonnacci suite

Utilities:

- sqrt
- 2nd degree

NAM-IP - gamma3/programs/fibo.txt -	Eclipse IDE										- 0	×
ile <u>E</u> dit <u>N</u> avigate Se <u>a</u> rch <u>P</u> roject	Run Window Help		ла — 🔚 — ич- и	* ^	-						○ :=> ■	<mark>19</mark>
			<u>אן א 8</u> 1 א געיק 			-	-	D		-	≪ :⊞ ®	••••••••••••••••••••••••••••••••••••••
Package Expl × 🗗 JUnit 🗆 🗆	🛃 testRun.java	🔊 BT.java 🛛 🔬 Con	nstants.java 🐰] Instruction	NOP.java	J SL.java	V.java	ibo.txt ×	6	🗒 Task List 🔋 Type Hierard	hy ×	
 Anstruction,java Instruction,java InstructionsParser,java IstructionsParser,java 	2 At each i 3 4 1A00 CD 5 1C07 CO 7 6 7 3200 ZB - 8 3300 ZB - 9 3100 ZB 10 A001 AN 11 8400 OB - 12 13 tant quest 4 15 6300 m1 = 16 6400 m1 = 17 8500 somm 18 19 6400 20 8300 m4 - 21 22 6500 23 8400 m5 - 24 8E00 ecri 25 26 6200	<pre>iteration M2 = r 7 n = 0 M2 u0 = 0 M3 u1 = 1 M4 : = u0 = u0 + u1 me dans M5 -> m3 -> m4 iture sur extract</pre>	n and M4 <u>fib</u>	(n)						'Instruction - instructions' - in ✓ Object ✓ AMD ④ AMD ④ AMD ④ BD ④ CB ④ CD ④ CD ④ CD ④ CO ④ CO ④ CSZ ④ CS ④ CS ④ CS ④ GG ④ CS ④ GG ⑤ GG ⑤ Operation ④ TT > ④ V ⑤ VCS ④ VRS	vorking set: Wi	ndow Wo
Machine RullCommo iovo	26 6200											
	28 8200 n++											
Connected Machine iava	298D00 ecri	iture sur extra	cteur sortie							G ^A Instruction	⊅ti ti 19	R R ⁵
	30									SgetChar(byte) : String		
> D Constants java	31 1800 affi	icher sortie 0								▲ AD		
	321801 affi	icher sortie 1								bullGamma		
> DrumTrack java	33 34 900E == com	nare n a 15									Fa	8 🗆
> DrumTrackGroup.java	35 011D V. S	saut 17								E Outline ×	θ'	8
> A Execution.iava		Jaco III							Ŧ	There is no active editor that	provides an out	tline.
> 📴 Group.java	4								Þ			
> 📝 MagneticDrum.java	🔝 Problems 🛛 @ Java	adoc 🛛 😥 Declaration	🔗 Search 📮 Co	onsole 🛛 📇	Git Staging 🛛 音 🛚 Syr	nchronize 👶 Ca	all Hierarchy			🔲 💥 💥 📑 🔂 🚍 🦉	🛯 🛃 🚍 🔻 📑	• -
> 🛺 > Memory.java	<terminated> testRun</terminated>	(1) [JUnit] C:\DEV\Eclip	se\eclipse-j2ee-20	22-03\plugins\c	org.eclipse.justj.ope	njdk.hotspot.jre.f	ull.win32.x86_64	_17.0.2.v20220201-1	08\jre\bin\javaw.exe	(5 févr. 2023, 08:24:21 - 08:24:	21) [pid: 12972]]
> 🔢 MemoryMode.java	8d00 OB	000000000005	0000000000	08								
> 🛃 Octad.java	1800 ESI 1801 ESI	0000000000005	00000000000	08								
> 🛂 Serie.java	900f CN	0000000000005	00000000000	08								
> 🛂 Series3.java	011d V	000000000005	0000000000	08								
> 🔏 Word.java	6300 BO	000000000005	0000000000	08								
✓	a400 AN	000000000005	0000000000	08								
> 🛂 TestAN.java	8500 OB	00000000013	0000000000	08								
> 🛂 testCN.java	8300 OB	000000000013	00000000000	08								
IestConsole.java	6500 BO	000000000008	00000000000	08								
> 🚜 lestDC.java	8400 OB	00000000013	0000000000	08								
InstMaching invo	8e00 OB	00000000013	0000000000	13								
	6200 BO	00000000013	0000000000	13								
restivientory.java	AUUI AN	000000000005	00000000000	13								

Some available FP subroutines

Opération	Durée opéra- tion	Variante	M ₁	M4	M ₅	M ₆	M ₇	M ₁	M4	M ₅	M ₆	
AR	1 pt	0.0 2 5	204		A	в		AB	inchan-	Α	AB	inchan-
D	1 pt		211	1	А	в	С	AB — C	inchan-	Α	AB	AB - C
B-C	2 pt	00 2 9			A	В	С	AB + C	inchan-	Α	AB	AB + C
$\mathbf{B} + \mathbf{C} \dots$	2 pt	0 0 2 13				A	в	A + B	inchan-	inchan-	А	A + B
+B	ı pt	00 7 13	1.8		manau	A	в	A — B	gée inchan-	inchan-	A	A — B
— В	ı pt	0071					B	A/B	gée inchan-	gée inchan-	A	A/B
/B	ı pt	0 0 15 18			x	A		A v2	gée	gée		
$Ax_2 + Bx + C$	4 pt	0 0 2 13	4	С		A	В	+ Bx + C	o chang.	x inchan-	o sin. x	O COS. X
$\begin{array}{c} \text{CD av. 4 col. 6)} \\ \text{in. et cos. } x \dots \end{array}$	4 cartes	en ligne 11			inchan- gée	x				gée		
rctg N/D	4 cartes		-	11215	inchan- gée	N	D	arctg	chang.	inchan- gée	chang.	arctg

Lessons Learned...

- Exploring early machines quite a strange journey:
 - Emerging concepts, still being explored
 - Gone technologies
 - ➔ need to look at larger historical context
- Not so complex to code but many details and limited original reference to test against
- Still a lot to explore (e.g. floating point), implement (UI...), gather/experiment with "code"
- In summary, an very rewarding experience both technically and culturally !



Questions ?



Some reference & credits

- Fédération des Equipes Bull for access to their GAMMA3 documentation at NAM-IP museum or online: <u>http://www.feb-patrimoine.com/projet/gamma3/gamma3.htm</u>
- Vincent Joguin for its DOS-based emulator
 - vidéo (french): <u>https://www.youtube.com/watch?v=X_ermLbQYLI</u>
 - executable: <u>http://vincent.joguin.com/GAMMAET.ZIP</u>
 - José Maillard and Lucas Trampal for the Open Source javascript emulator (coordinated by ACONIT) <u>https://github.com/lutrampal/bullgammator/</u>
- ACONIT for on-line documentation and running emulator and keeping the GAMMA3 memory alive with students
 - documentation: <u>https://www.aconit.org/histoire/Gamma-3</u>
 - online emulator: <u>https://www.aconit.org/histoire/Gamma-3/Simulateur</u>
- Code: <u>https://github.com/NAMIP-Computer-Museum/gamma3</u>

<u>Visit us</u>: <u>www.nam-ip.be</u> Twitter @ComputerMuseumB

<u>Contact me</u>: <u>christophe.ponsard@cetic.be</u> @cponsard @cponsard@ludosphere.fr