# QtRvSim

*Education from Assembly to Pipeline, Cache Performance, and C Level Programming*

Czech Technical University in Prague

**Jakub Dupák** ***dev@jakubdupak.com***
**Pavel Píša** ***pisa@fel.cvut.cz***
**Karel Kočí** ***cynerd@email.cz***
**Michal Štepanovský** ***michal.stepanovsky@gmail.com***

*comparch.edu.cvut.cz*

## QtRvSim Background

MipsIt

## QtRvSim Background

MipsIt

**QtMips** (2019)

- Karel Koci, Pavel Pisa

## QtRvSim Background

MipsIt

**QtMips** (2019)

- Karel Koci, Pavel Pisa

**QtRvSim** (2022)

- Jakub Dupak, Pavel Pisa, Max Hollmann
- GPL3
- Qt 5/6
- GitHub (cvut/qtrvsim)

# QtRvSim Background

MipsIt

**QtMips** (2019)

- Karel Koci, Pavel Pisa

**QtRvSim** (2022)

- Jakub Dupak, Pavel Pisa, Max Hollmann
- GPL3
- Qt 5/6
- GitHub (cvut/qtrvsim)

**CTU Computer Architecture Education**

- https://comparch.edu.cvut.cz

## QtRvSim Background

MipsIt

**QtMips** (2019)

- Karel Koci, Pavel Pisa

**QtRvSim** (2022)

- Jakub Dupak, Pavel Pisa, Max Hollmann
- GPL3
- Qt 5/6
- GitHub (cvut/qtrvsim)

**CTU Computer Architecture Education**

- https://comparch.edu.cvut.cz

**CTU RISC-V International Membership** (2023)

## QtRvSim: Usage in Teaching

**Czech Technical University in Prague**

- Faculty of Electrical Engineering
- Faculty of Information Technology

**Technical University in Graz**

**University of Colorado at Colorado Springs**

**University of Porto**

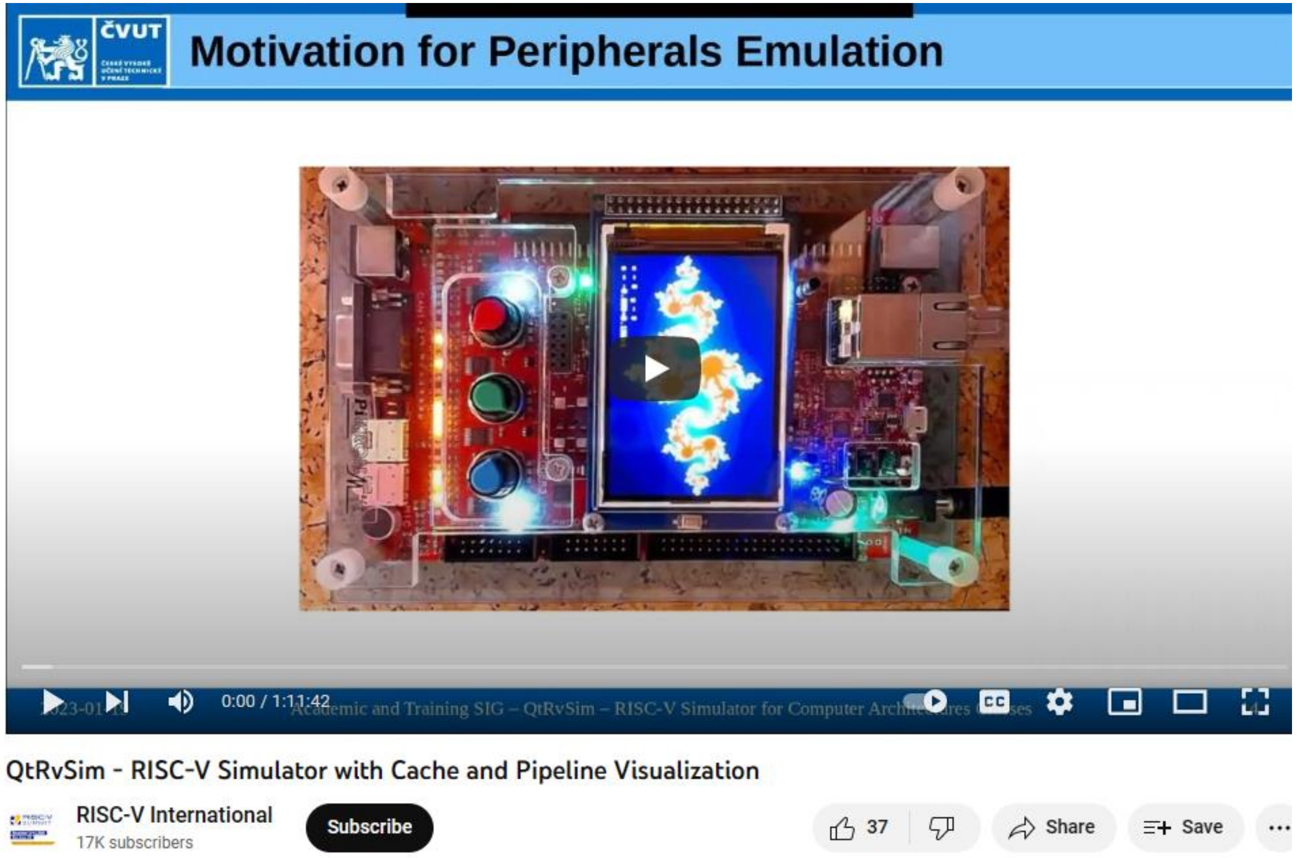
# QtMips: Usage in Teaching

**Charles University in Prague**

- Faculty of Mathematics and Physics

**National Kapodistrian University of Athens**

# Internal Design Overview



Motivation for Peripherals Emulation

0:00 / 1:11:42

QtRvSim - RISC-V Simulator with Cache and Pipeline Visualization

RISC-V International

17K subscribers

**Presented at RISCV.org Academia and Training SIG**

[https://youtu.be/J6AcPZZ_ISg](https://youtu.be/J6AcPZZ_ISg)

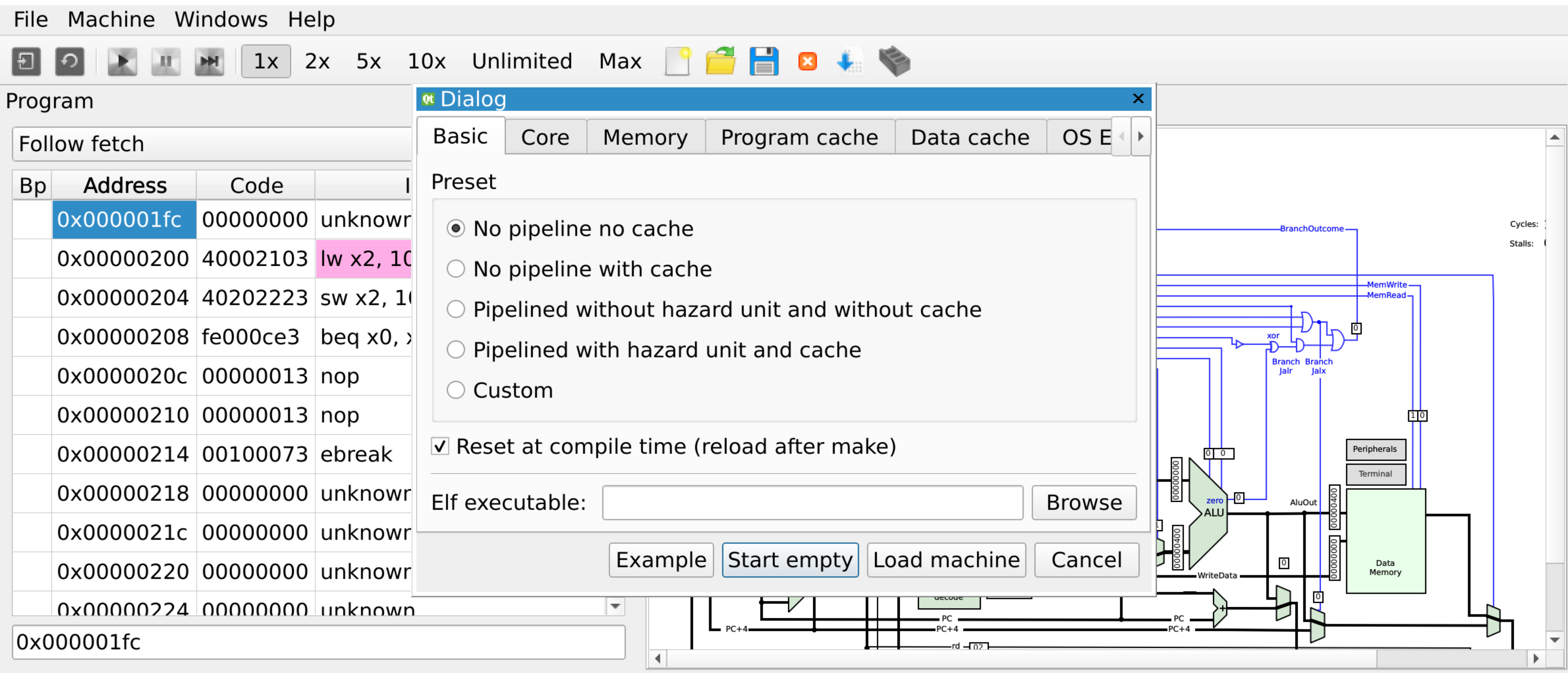
## WebAssembly Edition

comparch.edu.cvut.cz/qtrvsim/app

## First Steps In Assembly

Load Word
Store Word
Branch Equal

# Config



File Machine Windows Help

1x 2x 5x 10x Unlimited Max

Program

Follow fetch

| Bp | Address | Code | I |
|---|---|---|---|
| | 0x000001fc | 00000000 | unknowr |
| | 0x00000200 | 40002103 | lw x2, 10 |
| | 0x00000204 | 40202223 | sw x2, 1( |
| | 0x00000208 | fe000ce3 | beq x0, x |
| | 0x0000020c | 00000013 | nop |
| | 0x00000210 | 00000013 | nop |
| | 0x00000214 | 00100073 | ebreak |
| | 0x00000218 | 00000000 | unknowr |
| | 0x0000021c | 00000000 | unknowr |
| | 0x00000220 | 00000000 | unknowr |
| | 0x00000224 | 00000000 | unknown |

0x000001fc

Dialog

Basic | Core | Memory | Program cache | Data cache | OS E

Preset

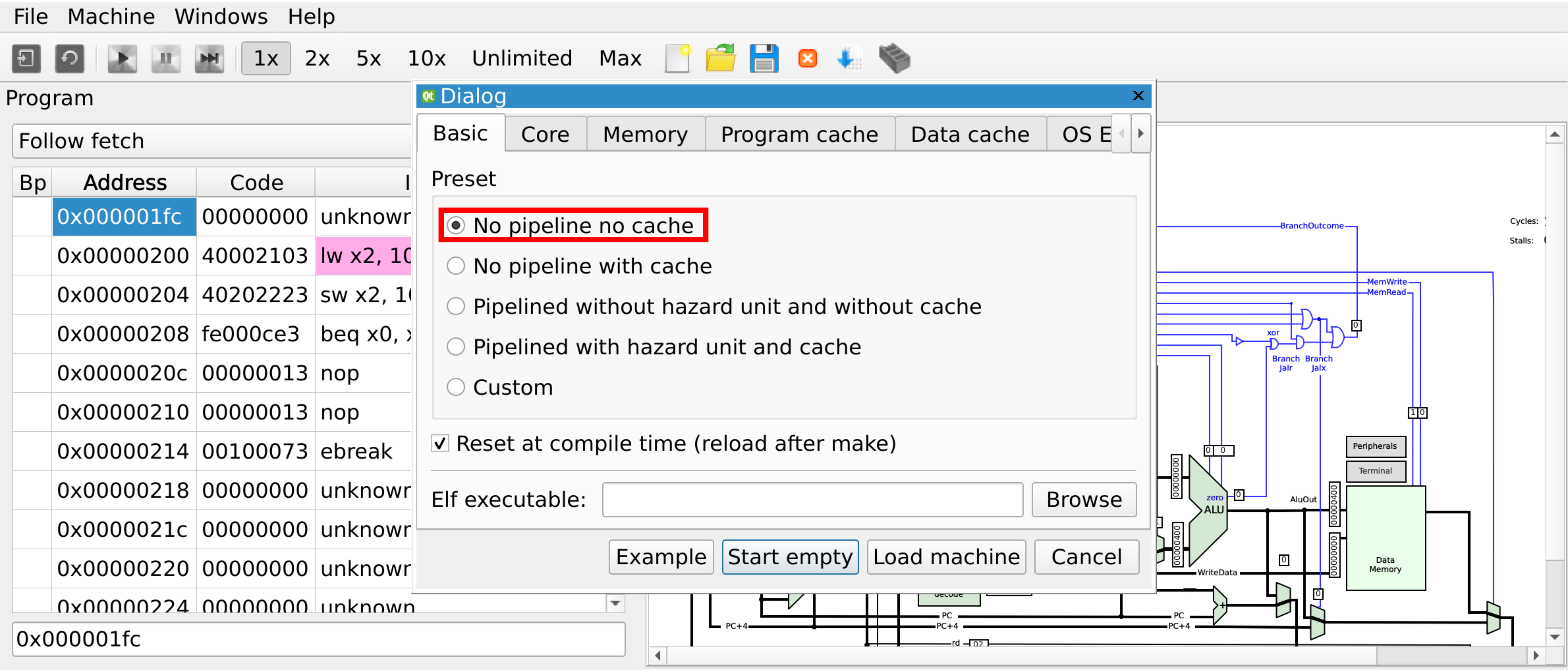
- (●) No pipeline no cache
- ( ) No pipeline with cache
- ( ) Pipelined without hazard unit and without cache
- ( ) Pipelined with hazard unit and cache
- ( ) Custom

✓ Reset at compile time (reload after make)

Elf executable: Browse

Example | Start empty | Load machine | Cancel

# Config


File Machine Windows Help
1x 2x 5x 10x Unlimited Max
Program
Follow fetch
Dialog
Basic Core Memory Program cache Data cache OS E
Browse
Example Start empty Load machine Cancel

Preset

- [✓ No pipeline no cache]
- No pipeline with cache
- Pipelined without hazard unit and without cache
- Pipelined with hazard unit and cache
- Custom

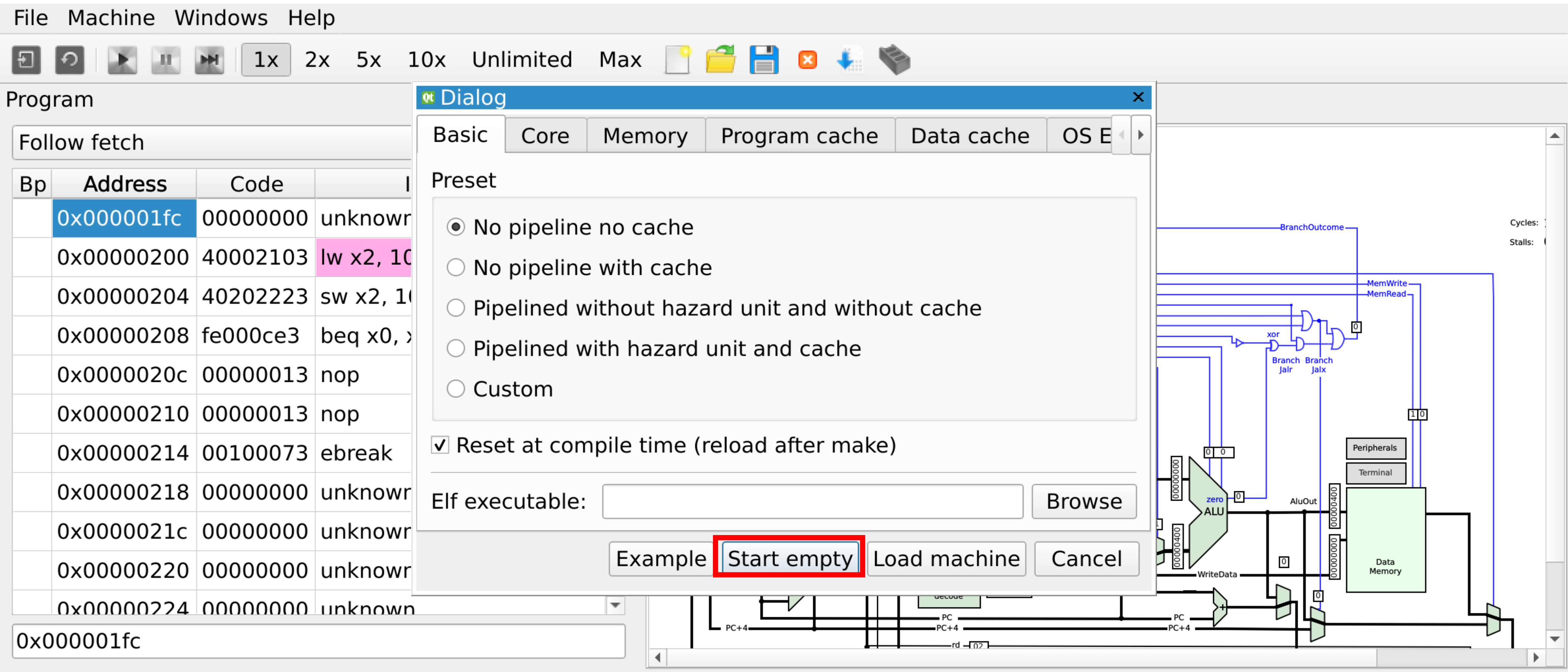
✓ Reset at compile time (reload after make)

Elf executable:

| Bp | Address | Code | I |
|---|---|---|---|
| | 0x000001fc | 00000000 | unknowr |
| | 0x00000200 | 40002103 | lw x2, 10 |
| | 0x00000204 | 40202223 | sw x2, 1( |
| | 0x00000208 | fe000ce3 | beq x0, x |
| | 0x0000020c | 00000013 | nop |
| | 0x00000210 | 00000013 | nop |
| | 0x00000214 | 00100073 | ebreak |
| | 0x00000218 | 00000000 | unknowr |
| | 0x0000021c | 00000000 | unknowr |
| | 0x00000220 | 00000000 | unknowr |
| | 0x00000224 | 00000000 | unknown |

0x000001fc

# Config



| Bp | Address | Code | I |
|---|---|---|---|
| | 0x000001fc | 00000000 | unknowr |
| | 0x00000200 | 40002103 | lw x2, 10 |
| | 0x00000204 | 40202223 | sw x2, 1( |
| | 0x00000208 | fe000ce3 | beq x0, x |
| | 0x0000020c | 00000013 | nop |
| | 0x00000210 | 00000013 | nop |
| | 0x00000214 | 00100073 | ebreak |
| | 0x00000218 | 00000000 | unknowr |
| | 0x0000021c | 00000000 | unknowr |
| | 0x00000220 | 00000000 | unknowr |
| | 0x00000224 | 00000000 | unknown |

0x000001fc

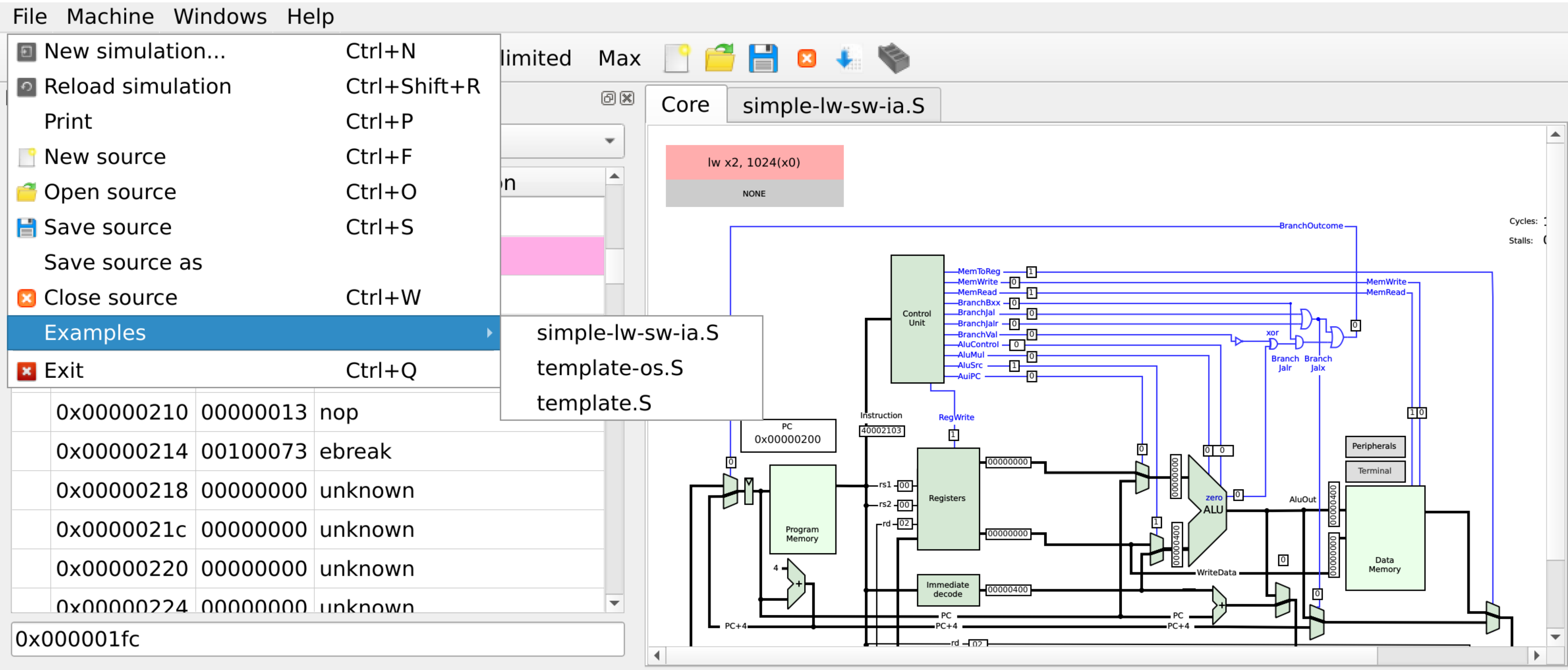
Preset

- ● No pipeline no cache
- ○ No pipeline with cache
- ○ Pipelined without hazard unit and without cache
- ○ Pipelined with hazard unit and cache
- ○ Custom

✓ Reset at compile time (reload after make)

Elf executable:

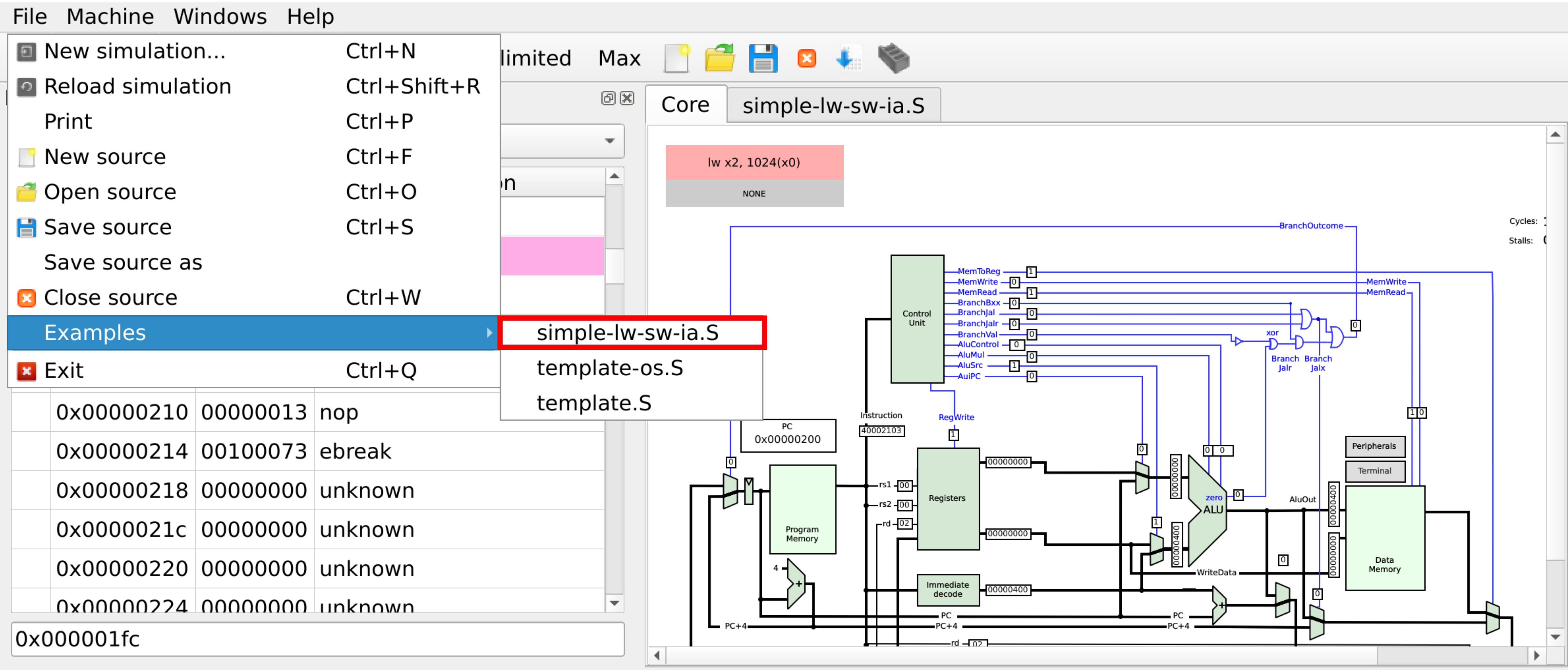
# Examples



```
0x00000210  00000013  nop
0x00000214  00100073  ebreak
0x00000218  00000000  unknown
0x0000021c  00000000  unknown
0x00000220  00000000  unknown
0x00000224  00000000  unknown
```

0x000001fc

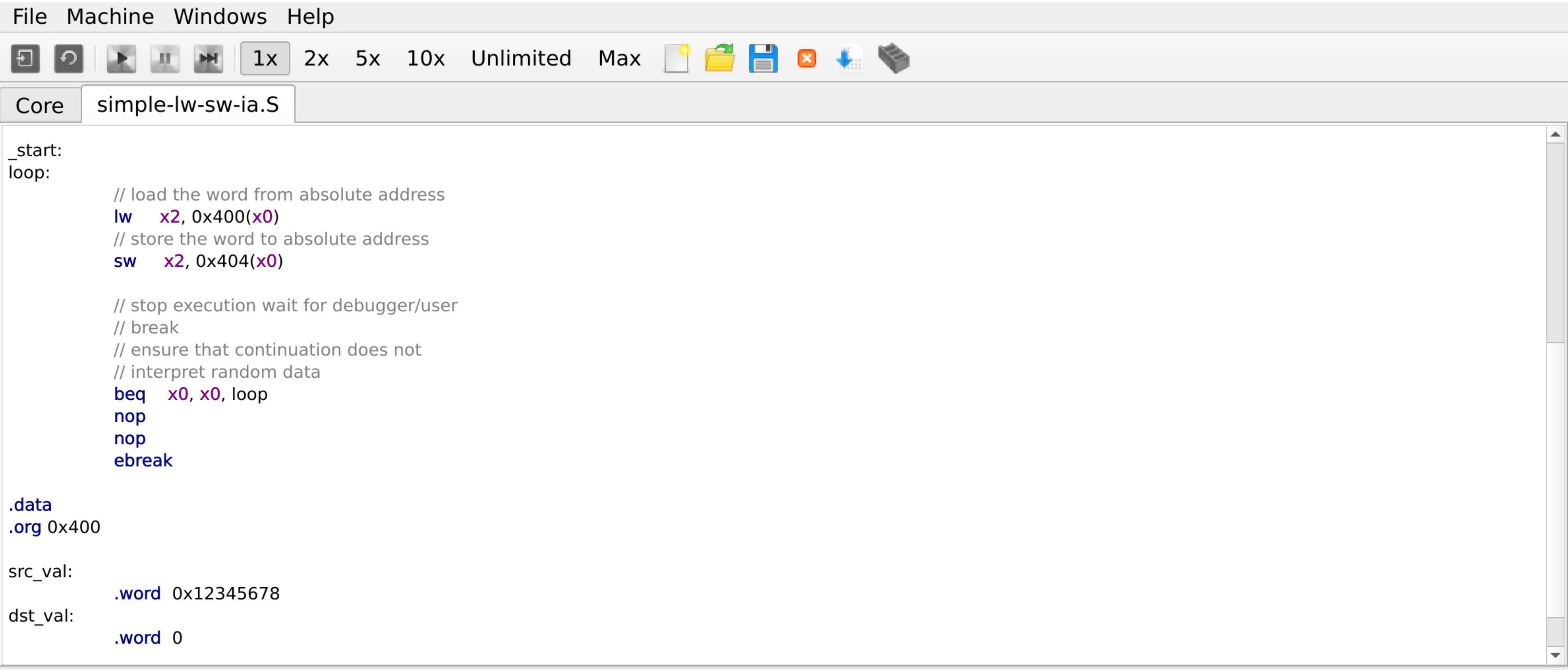
# Examples



```
0x00000210  00000013  nop
0x00000214  00100073  ebreak
0x00000218  00000000  unknown
0x0000021c  00000000  unknown
0x00000220  00000000  unknown
0x00000224  00000000  unknown
```

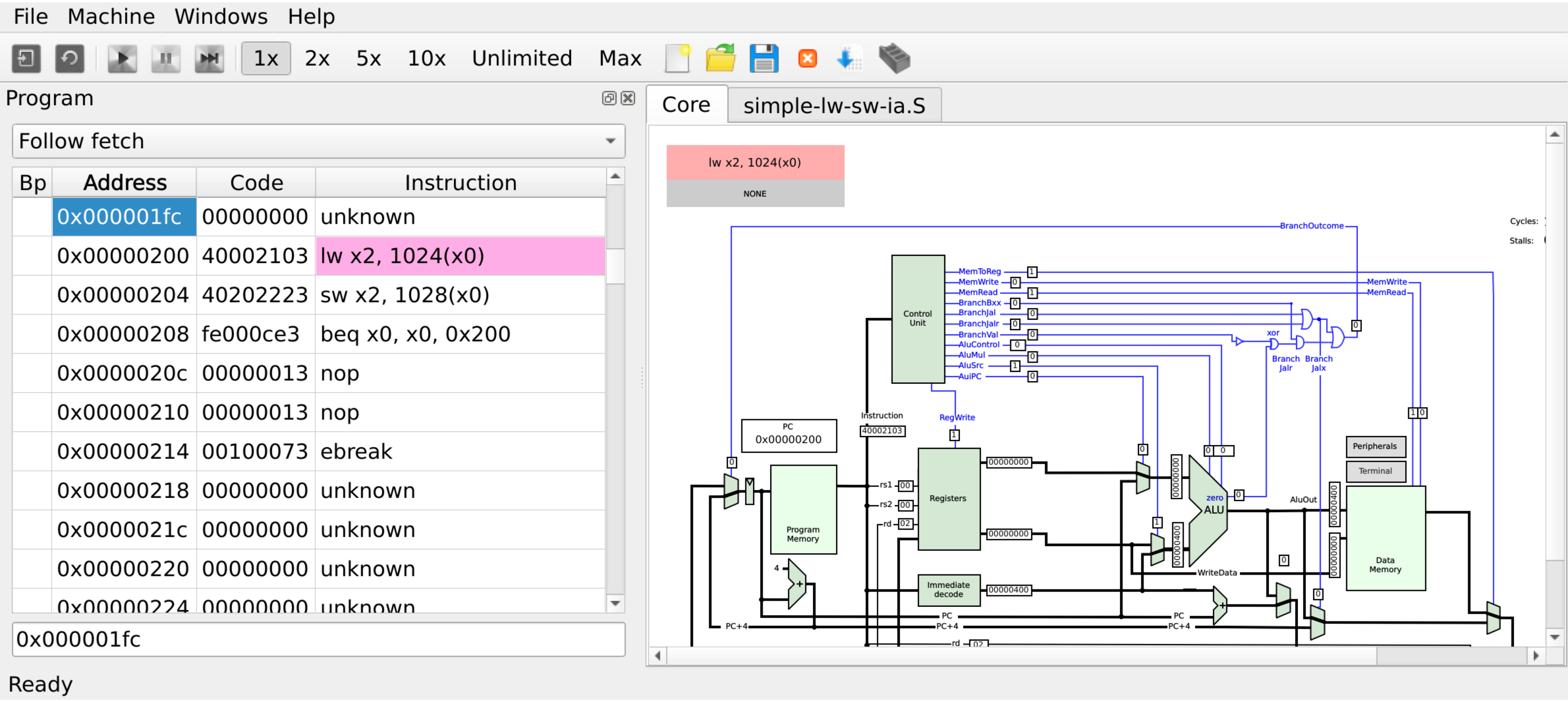
0x000001fc

# Example Code



```
_start:
loop:
            // load the word from absolute address
            lw    x2, 0x400(x0)
            // store the word to absolute address
            sw    x2, 0x404(x0)

            // stop execution wait for debugger/user
            // break
            // ensure that continuation does not
            // interpret random data
            beq   x0, x0, loop
            nop
            nop
            ebreak

.data
.org 0x400

src_val:
            .word  0x12345678
dst_val:
            .word  0
```

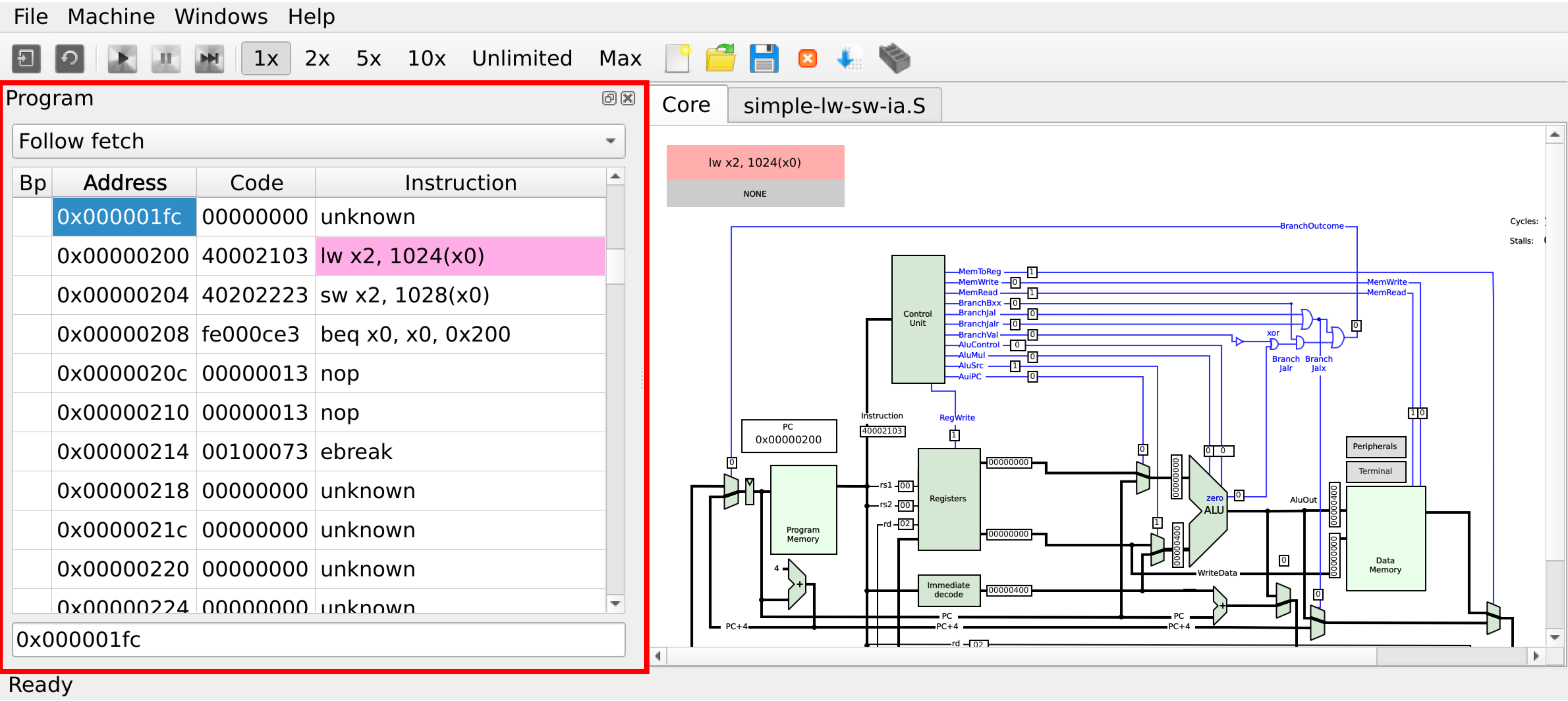
# Basic View



| Bp | Address | Code | Instruction |
|---|---|---|---|
| | 0x000001fc | 00000000 | unknown |
| | 0x00000200 | 40002103 | lw x2, 1024(x0) |
| | 0x00000204 | 40202223 | sw x2, 1028(x0) |
| | 0x00000208 | fe000ce3 | beq x0, x0, 0x200 |
| | 0x0000020c | 00000013 | nop |
| | 0x00000210 | 00000013 | nop |
| | 0x00000214 | 00100073 | ebreak |
| | 0x00000218 | 00000000 | unknown |
| | 0x0000021c | 00000000 | unknown |
| | 0x00000220 | 00000000 | unknown |
| | 0x00000224 | 00000000 | unknown |

0x000001fc

# Basic View



Program

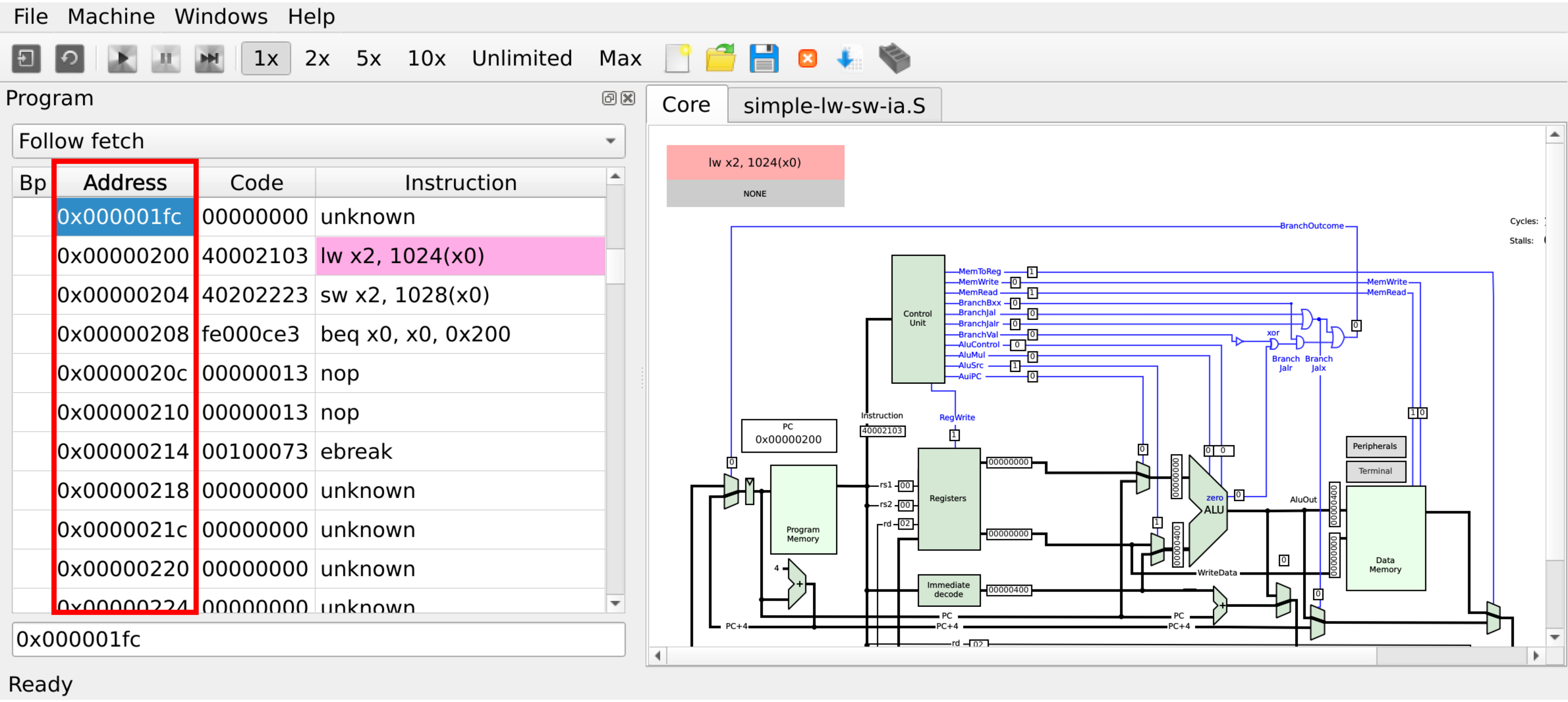
Follow fetch

| Bp | Address | Code | Instruction |
|---|---|---|---|
| | 0x000001fc | 00000000 | unknown |
| | 0x00000200 | 40002103 | lw x2, 1024(x0) |
| | 0x00000204 | 40202223 | sw x2, 1028(x0) |
| | 0x00000208 | fe000ce3 | beq x0, x0, 0x200 |
| | 0x0000020c | 00000013 | nop |
| | 0x00000210 | 00000013 | nop |
| | 0x00000214 | 00100073 | ebreak |
| | 0x00000218 | 00000000 | unknown |
| | 0x0000021c | 00000000 | unknown |
| | 0x00000220 | 00000000 | unknown |
| | 0x00000224 | 00000000 | unknown |

0x000001fc

Ready

# Basic View



File Machine Windows Help

1x 2x 5x 10x Unlimited Max

Program

Follow fetch

| Bp | Address | Code | Instruction |
|---|---|---|---|
| | 0x000001fc | 00000000 | unknown |
| | 0x00000200 | 40002103 | lw x2, 1024(x0) |
| | 0x00000204 | 40202223 | sw x2, 1028(x0) |
| | 0x00000208 | fe000ce3 | beq x0, x0, 0x200 |
| | 0x0000020c | 00000013 | nop |
| | 0x00000210 | 00000013 | nop |
| | 0x00000214 | 00100073 | ebreak |
| | 0x00000218 | 00000000 | unknown |
| | 0x0000021c | 00000000 | unknown |
| | 0x00000220 | 00000000 | unknown |
| | 0x00000224 | 00000000 | unknown |

0x000001fc

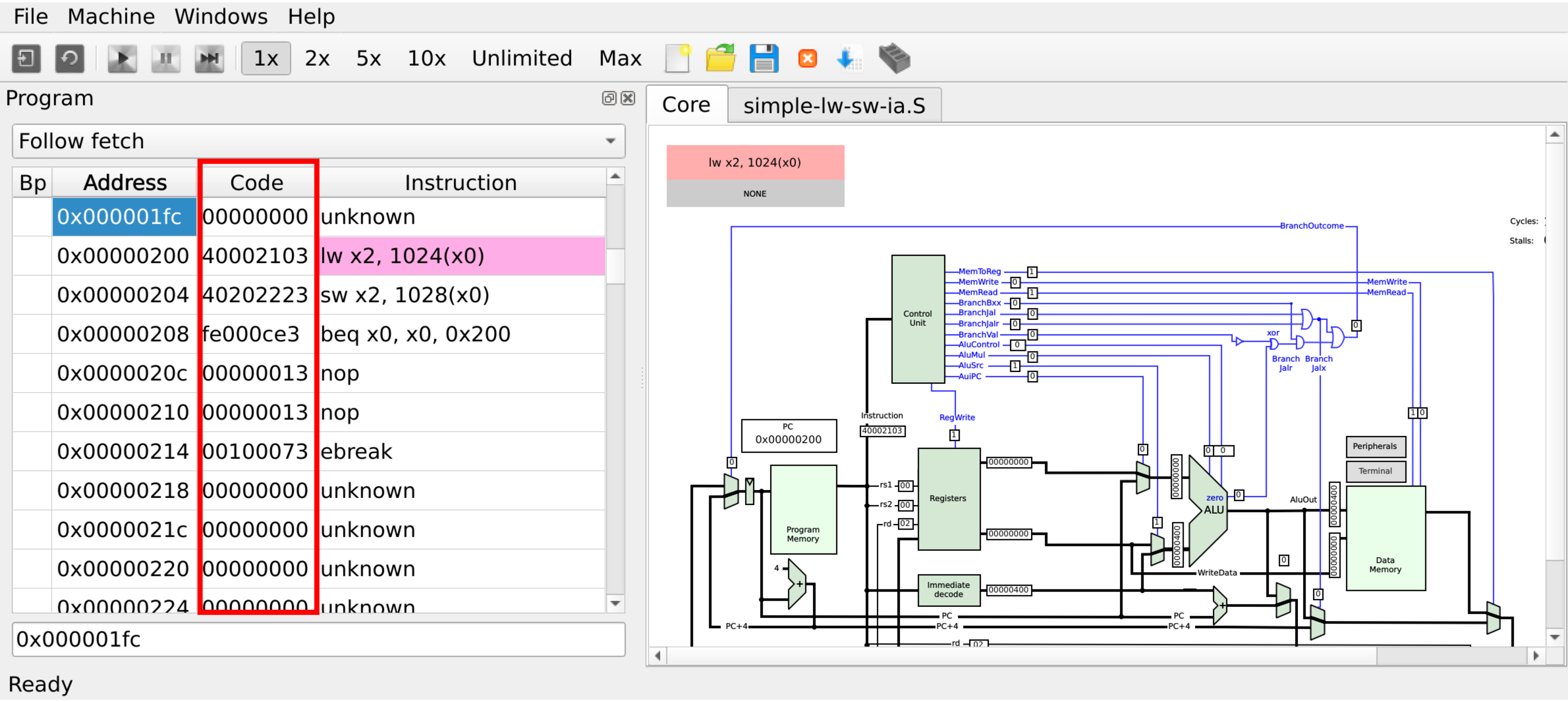
Ready

Core simple-lw-sw-ia.S

lw x2, 1024(x0)

NONE

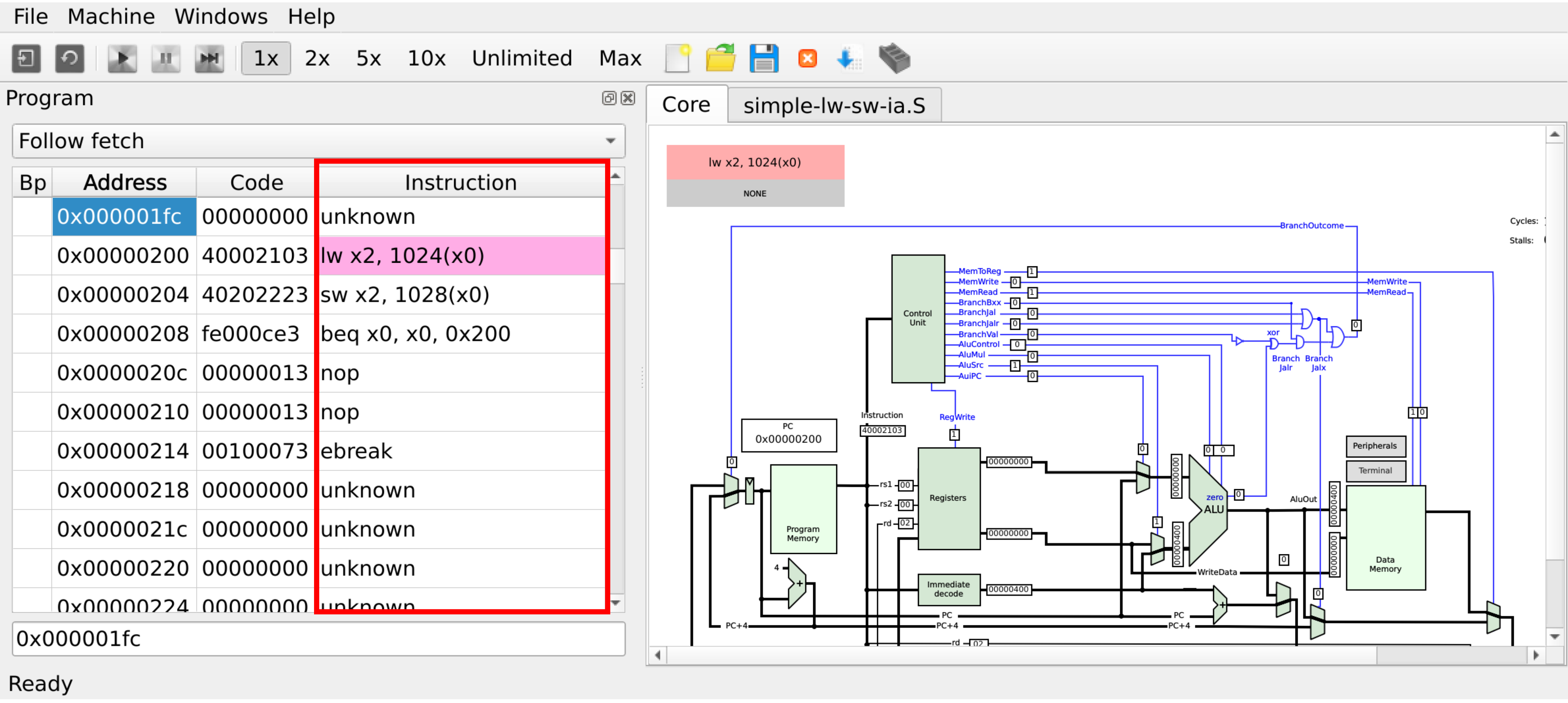
# Basic View



| Bp | Address | Code | Instruction |
|---|---|---|---|
| | 0x000001fc | 00000000 | unknown |
| | 0x00000200 | 40002103 | lw x2, 1024(x0) |
| | 0x00000204 | 40202223 | sw x2, 1028(x0) |
| | 0x00000208 | fe000ce3 | beq x0, x0, 0x200 |
| | 0x0000020c | 00000013 | nop |
| | 0x00000210 | 00000013 | nop |
| | 0x00000214 | 00100073 | ebreak |
| | 0x00000218 | 00000000 | unknown |
| | 0x0000021c | 00000000 | unknown |
| | 0x00000220 | 00000000 | unknown |
| | 0x00000224 | 00000000 | unknown |

0x000001fc

# Basic View

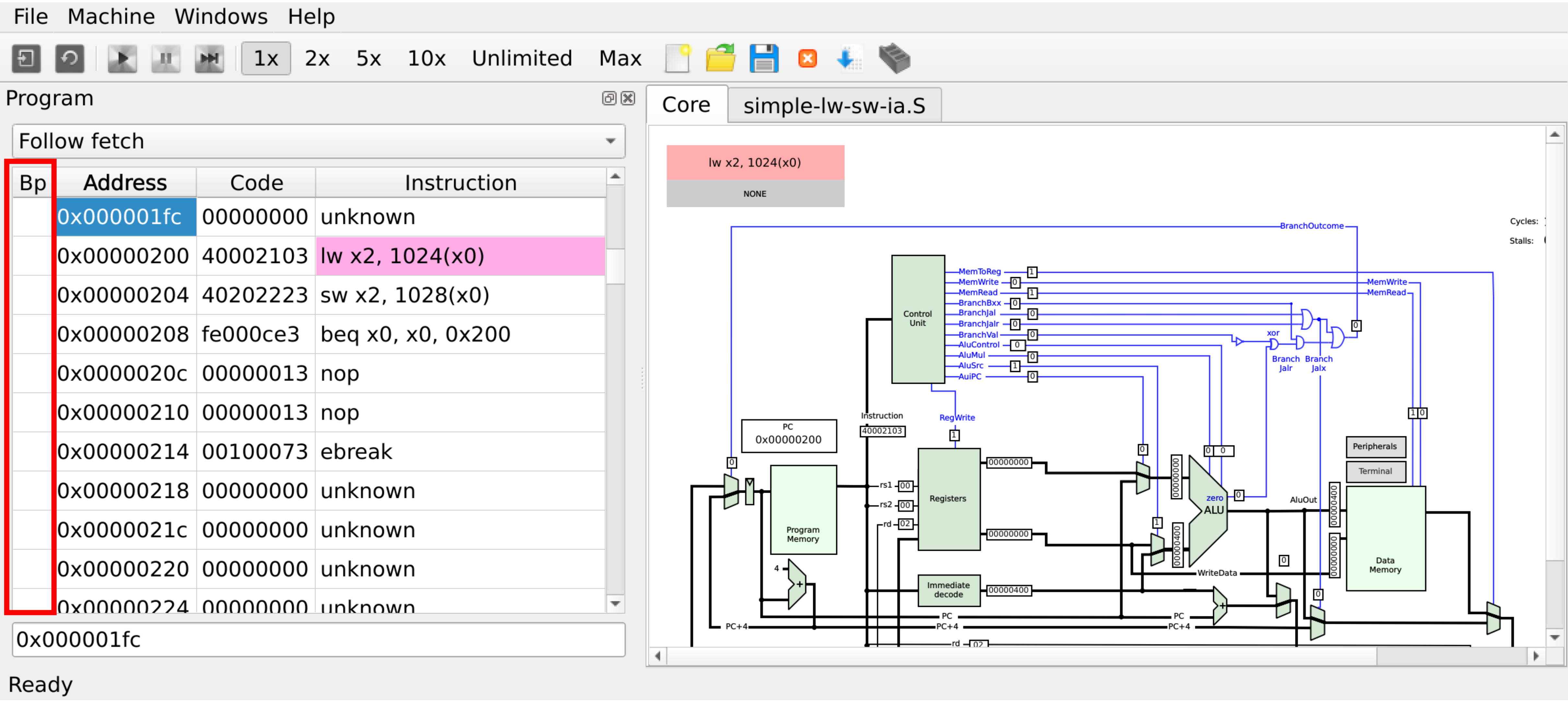


| Bp | Address | Code | Instruction |
|---|---|---|---|
| | 0x000001fc | 00000000 | unknown |
| | 0x00000200 | 40002103 | lw x2, 1024(x0) |
| | 0x00000204 | 40202223 | sw x2, 1028(x0) |
| | 0x00000208 | fe000ce3 | beq x0, x0, 0x200 |
| | 0x0000020c | 00000013 | nop |
| | 0x00000210 | 00000013 | nop |
| | 0x00000214 | 00100073 | ebreak |
| | 0x00000218 | 00000000 | unknown |
| | 0x0000021c | 00000000 | unknown |
| | 0x00000220 | 00000000 | unknown |
| | 0x00000224 | 00000000 | unknown |

0x000001fc

Ready

# Basic View



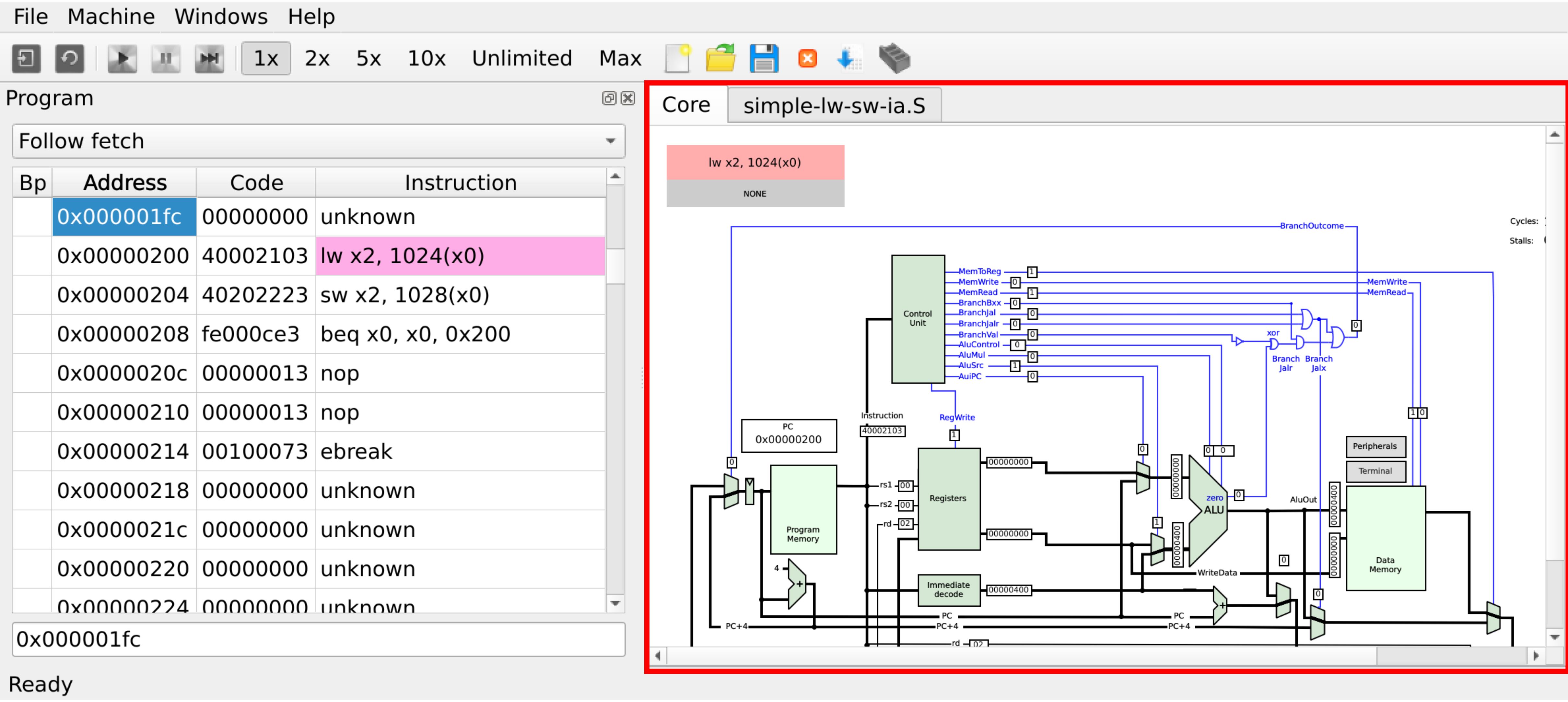
Follow fetch

| Bp | Address | Code | Instruction |
|---|---|---|---|
| | 0x000001fc | 00000000 | unknown |
| | 0x00000200 | 40002103 | lw x2, 1024(x0) |
| | 0x00000204 | 40202223 | sw x2, 1028(x0) |
| | 0x00000208 | fe000ce3 | beq x0, x0, 0x200 |
| | 0x0000020c | 00000013 | nop |
| | 0x00000210 | 00000013 | nop |
| | 0x00000214 | 00100073 | ebreak |
| | 0x00000218 | 00000000 | unknown |
| | 0x0000021c | 00000000 | unknown |
| | 0x00000220 | 00000000 | unknown |
| | 0x00000224 | 00000000 | unknown |

0x000001fc

Ready

# Basic View



File Machine Windows Help

1x 2x 5x 10x Unlimited Max

Program

Follow fetch

| Bp | Address | Code | Instruction |
|---|---|---|---|
| | 0x000001fc | 00000000 | unknown |
| | 0x00000200 | 40002103 | lw x2, 1024(x0) |
| | 0x00000204 | 40202223 | sw x2, 1028(x0) |
| | 0x00000208 | fe000ce3 | beq x0, x0, 0x200 |
| | 0x0000020c | 00000013 | nop |
| | 0x00000210 | 00000013 | nop |
| | 0x00000214 | 00100073 | ebreak |
| | 0x00000218 | 00000000 | unknown |
| | 0x0000021c | 00000000 | unknown |
| | 0x00000220 | 00000000 | unknown |
| | 0x00000224 | 00000000 | unknown |

0x000001fc

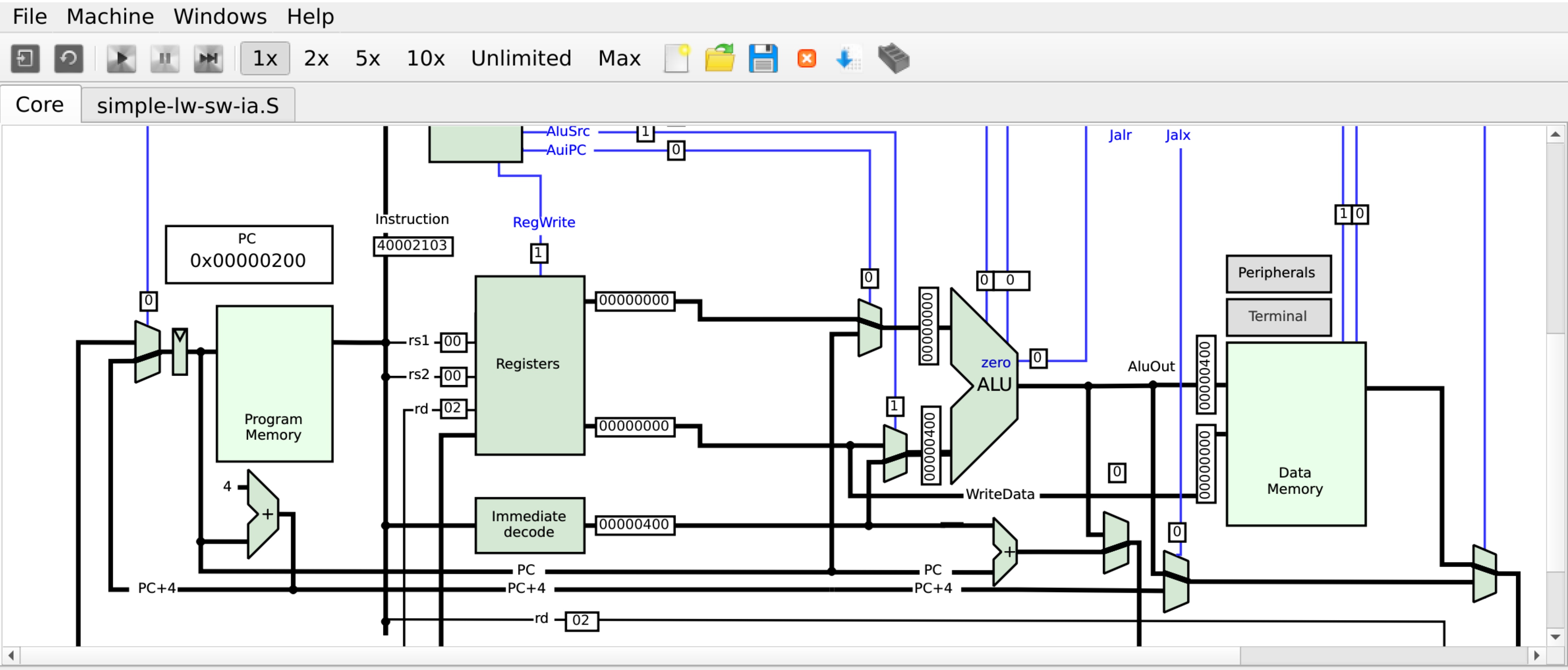
Ready

Core simple-lw-sw-ia.S

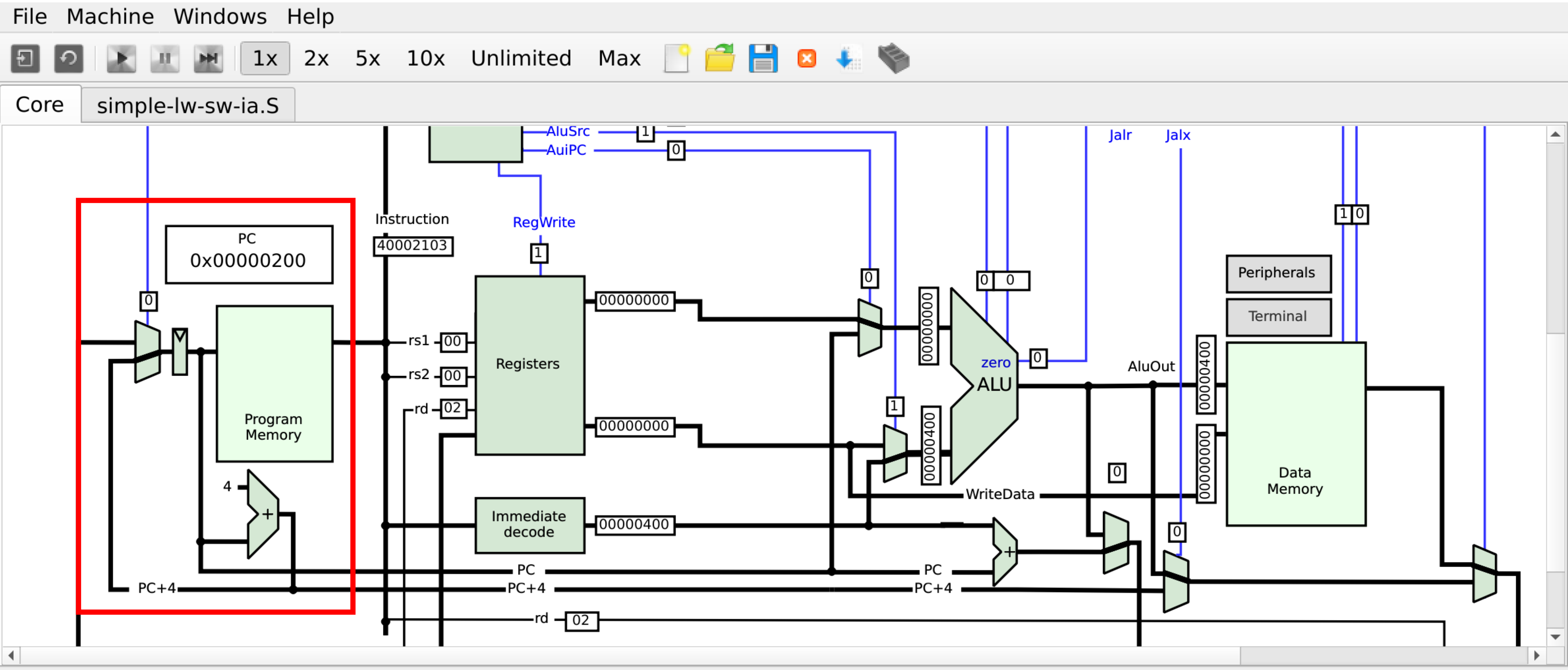
lw x2, 1024(x0)

NONE

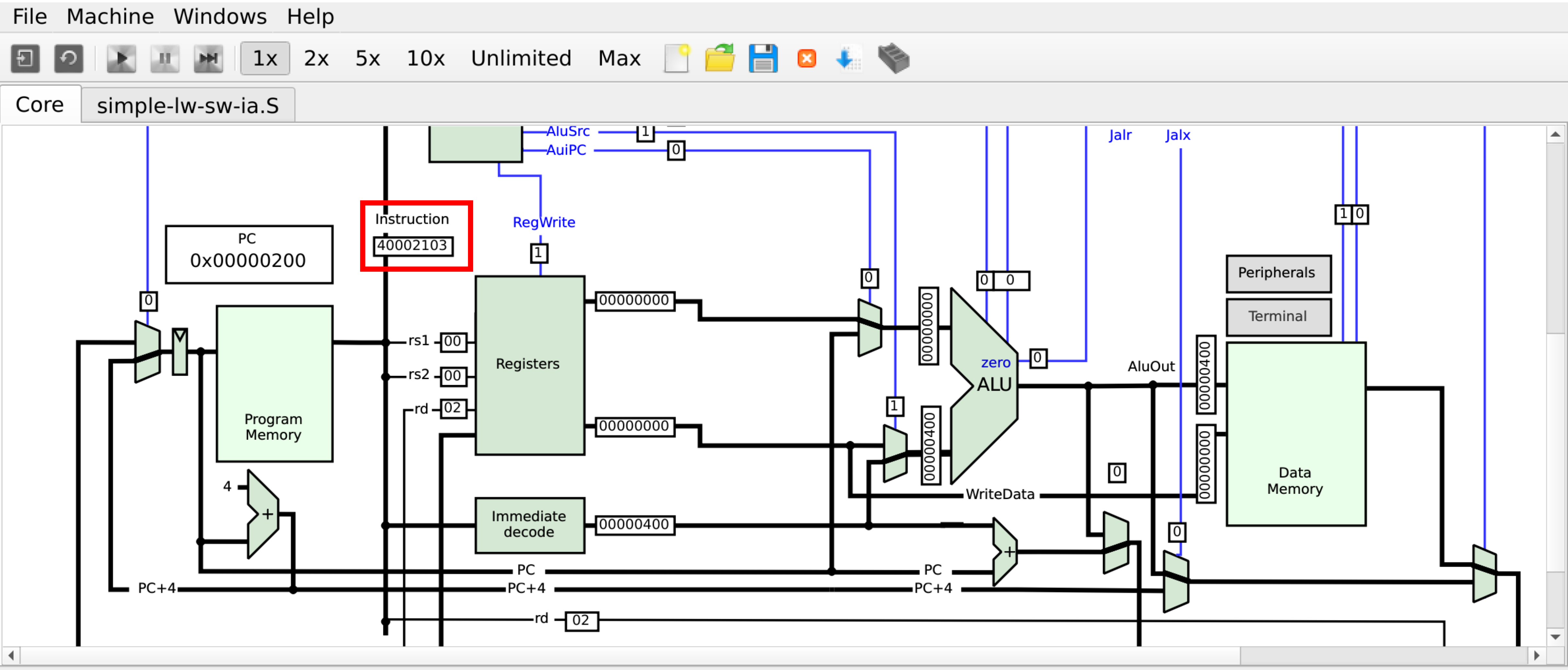
# Core View Detail: PC & Program Memory



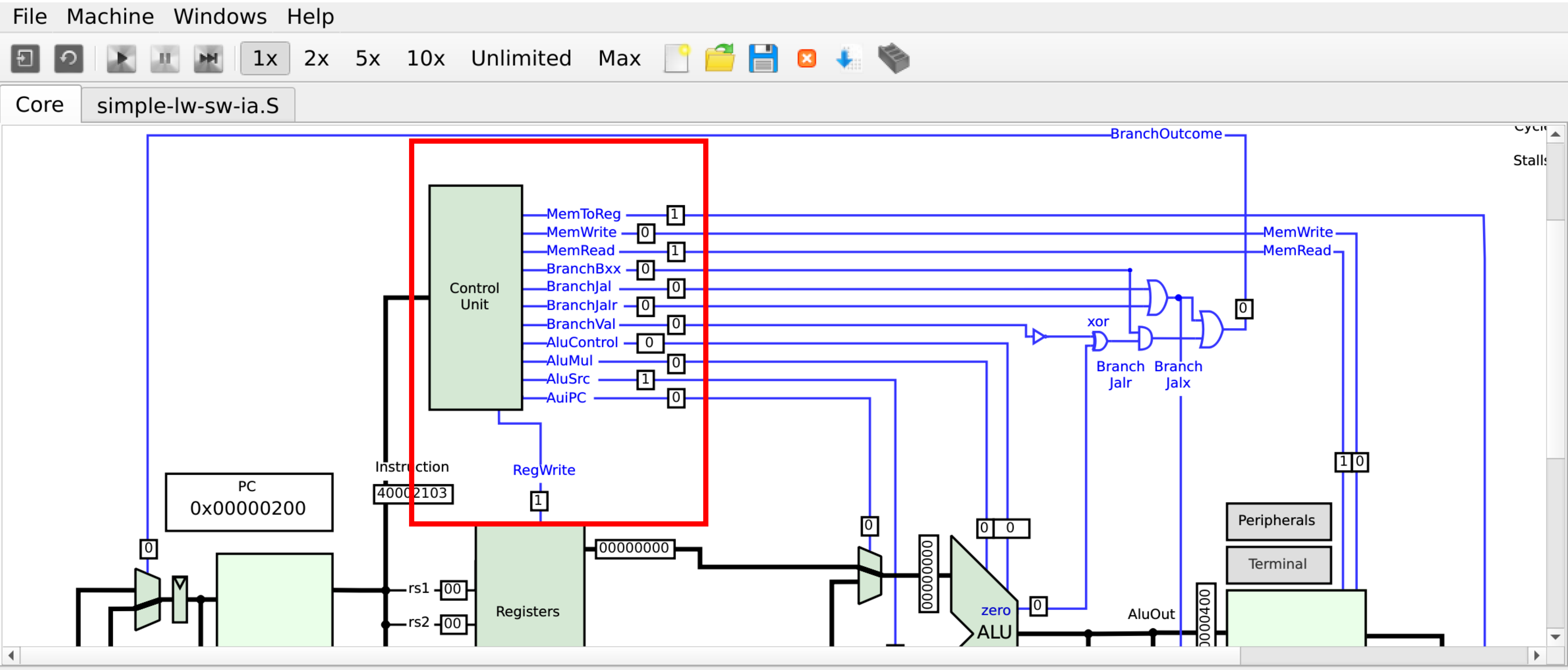
# Core View Detail: PC & Program Memory



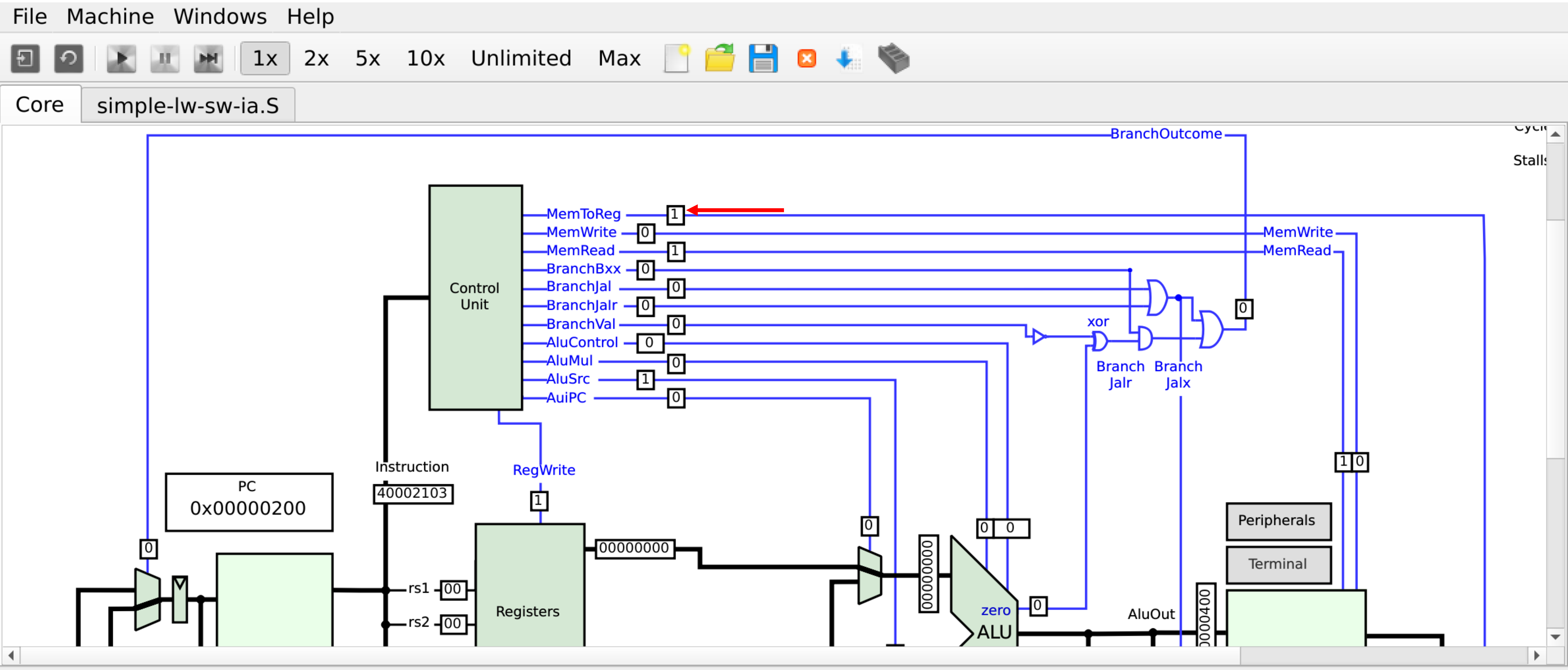
# Core View Detail: PC & Program Memory



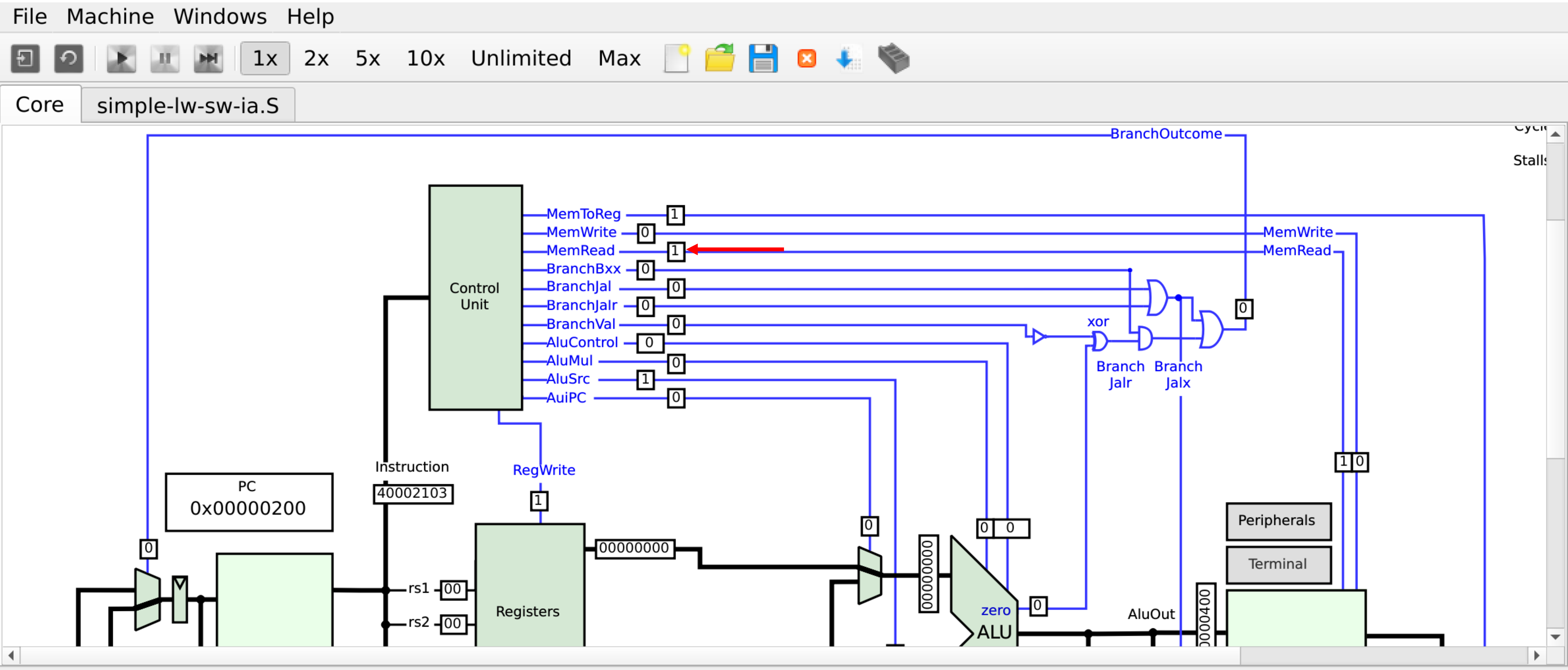
# Core View Detail: Control Signals



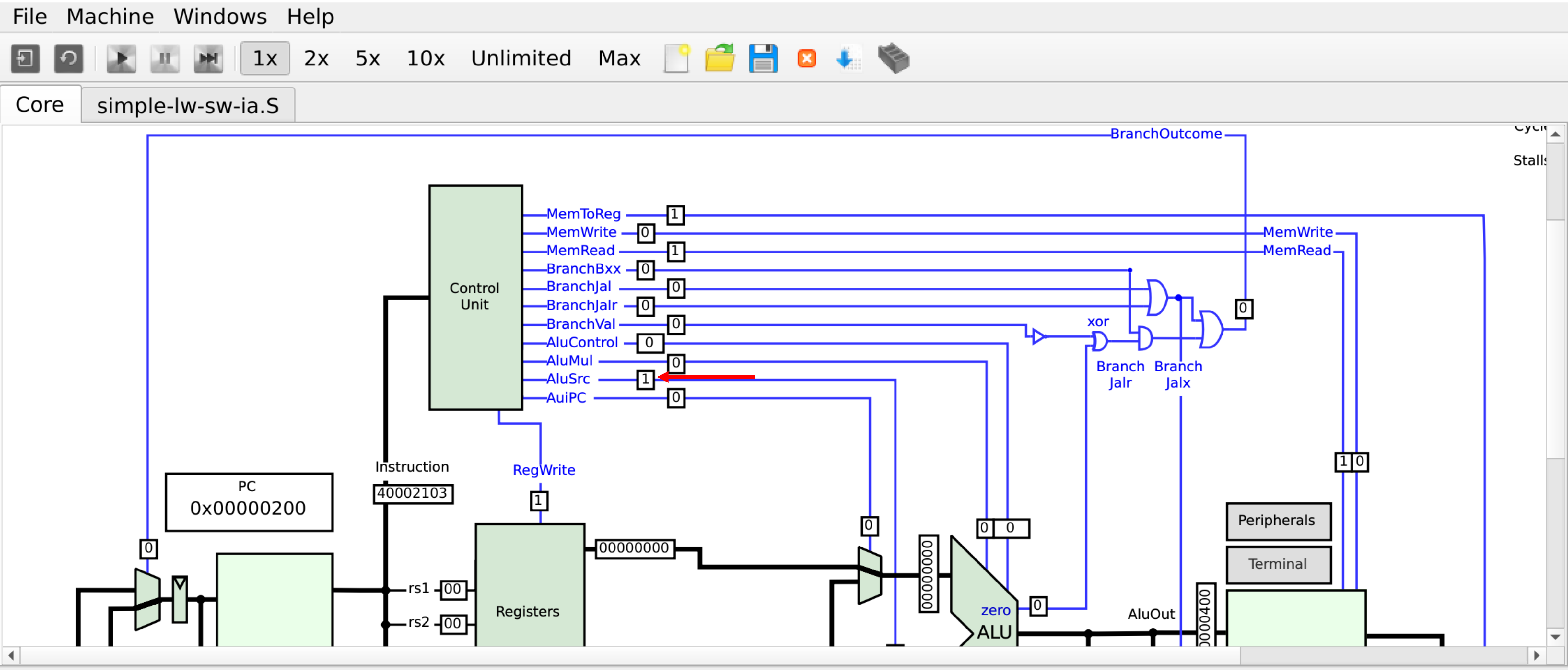
# Core View Detail: Control Signals



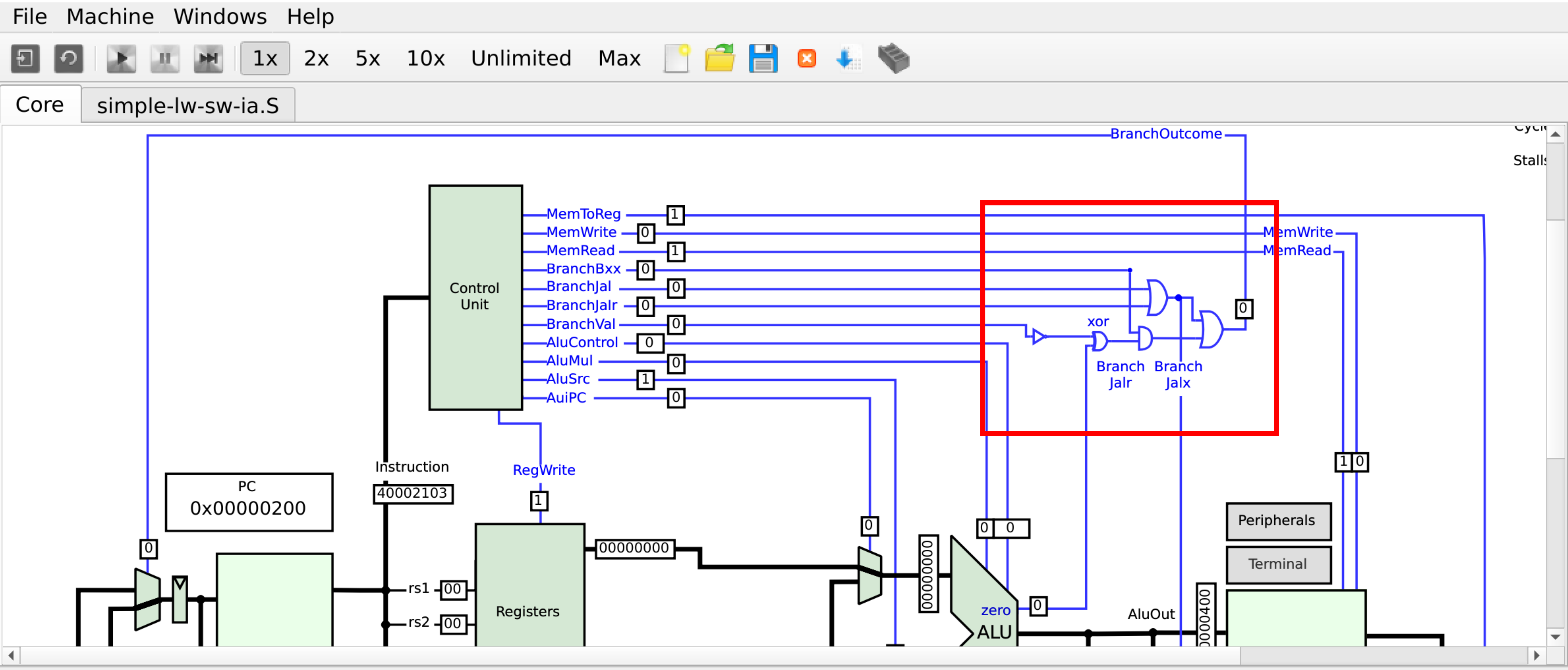
# Core View Detail: Control Signals



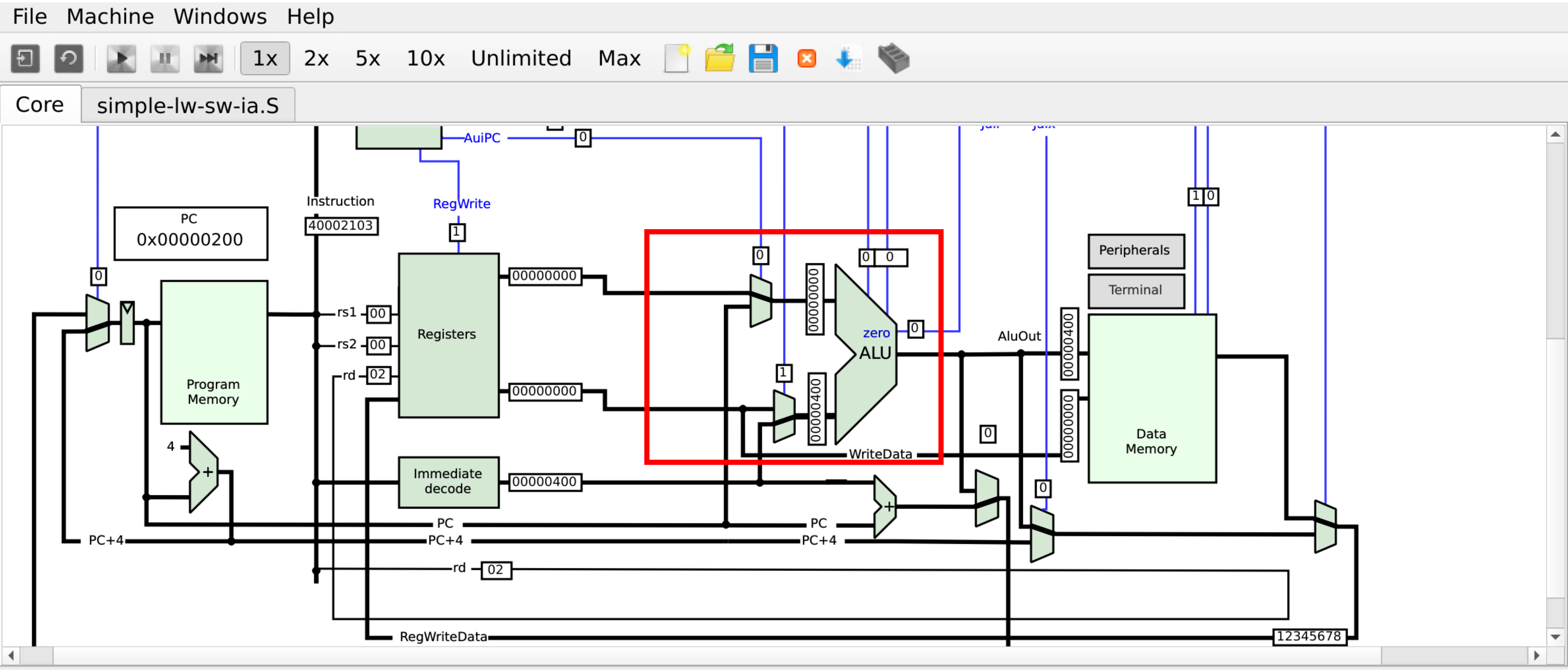
# Core View Detail: Control Signals



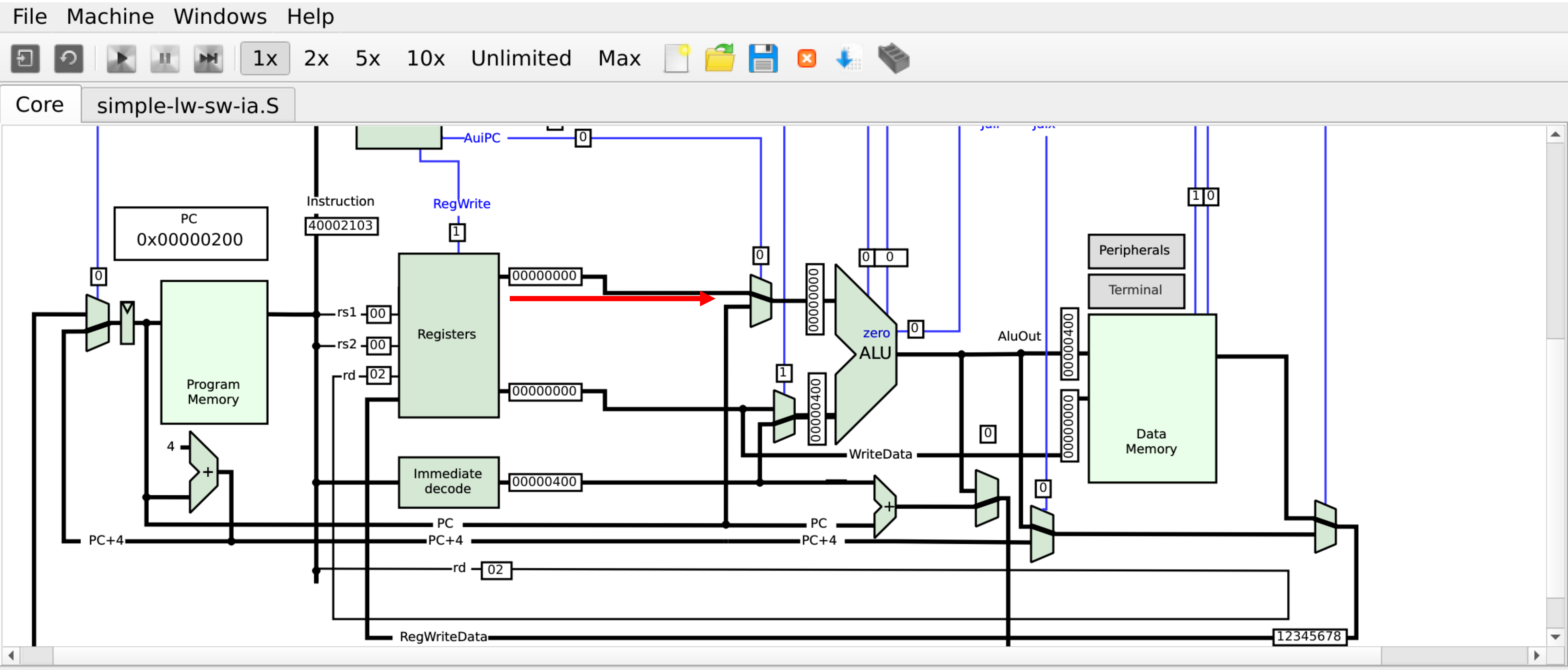
# Core View Detail: Control Signals



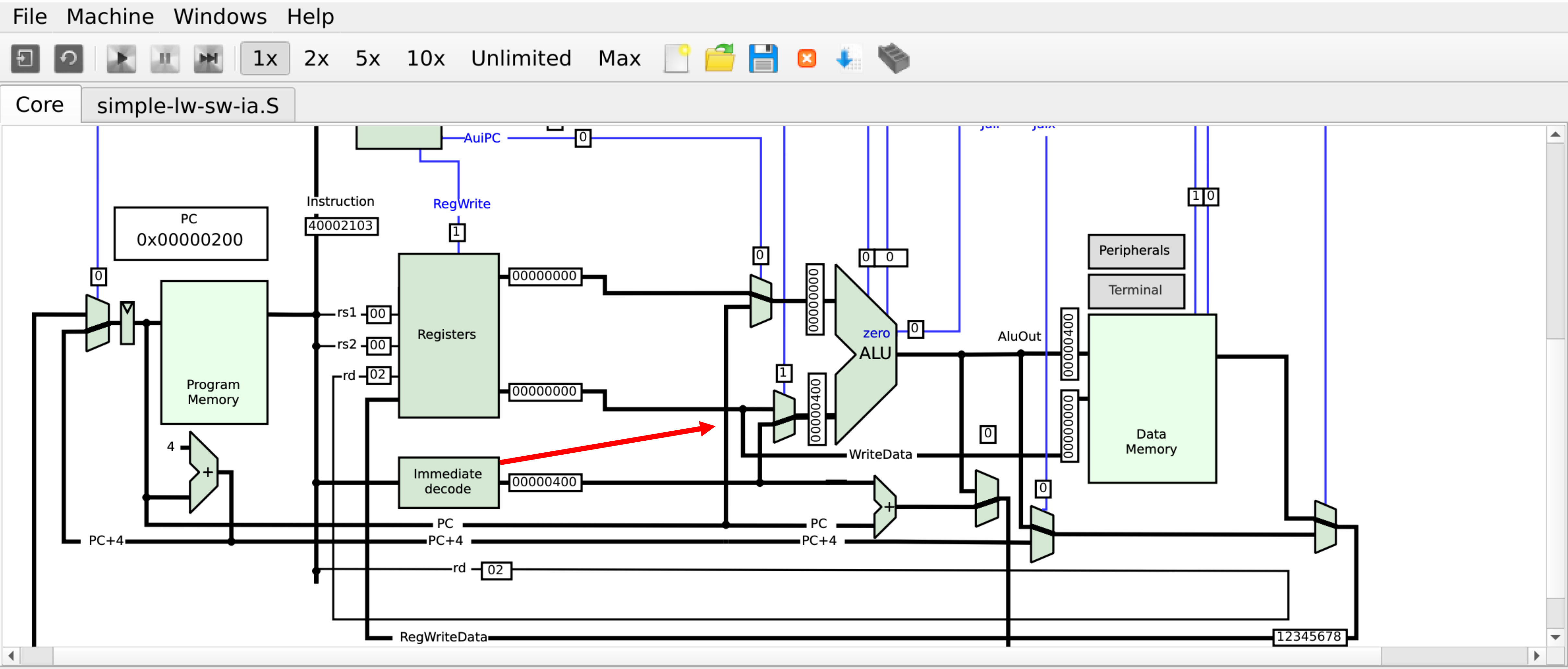
# Core View Detail: Execution



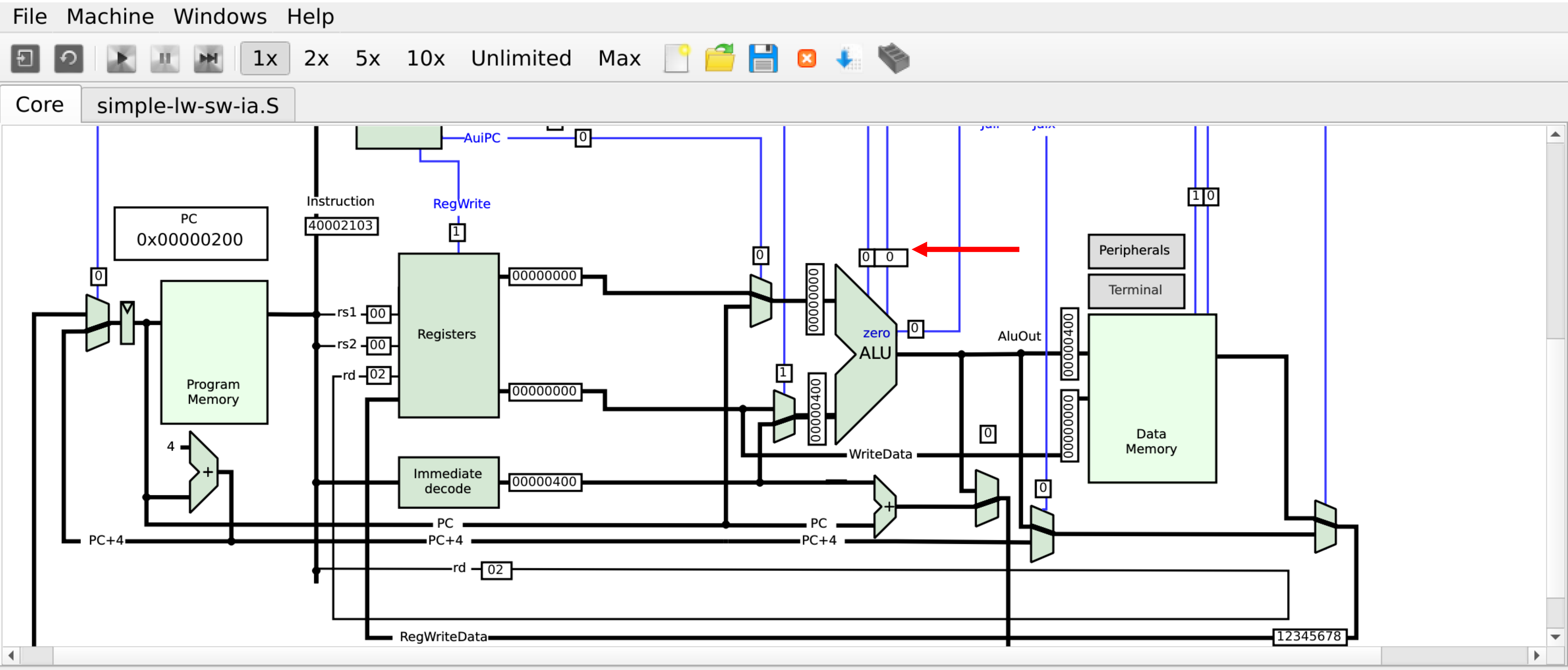
# Core View Detail: Execution



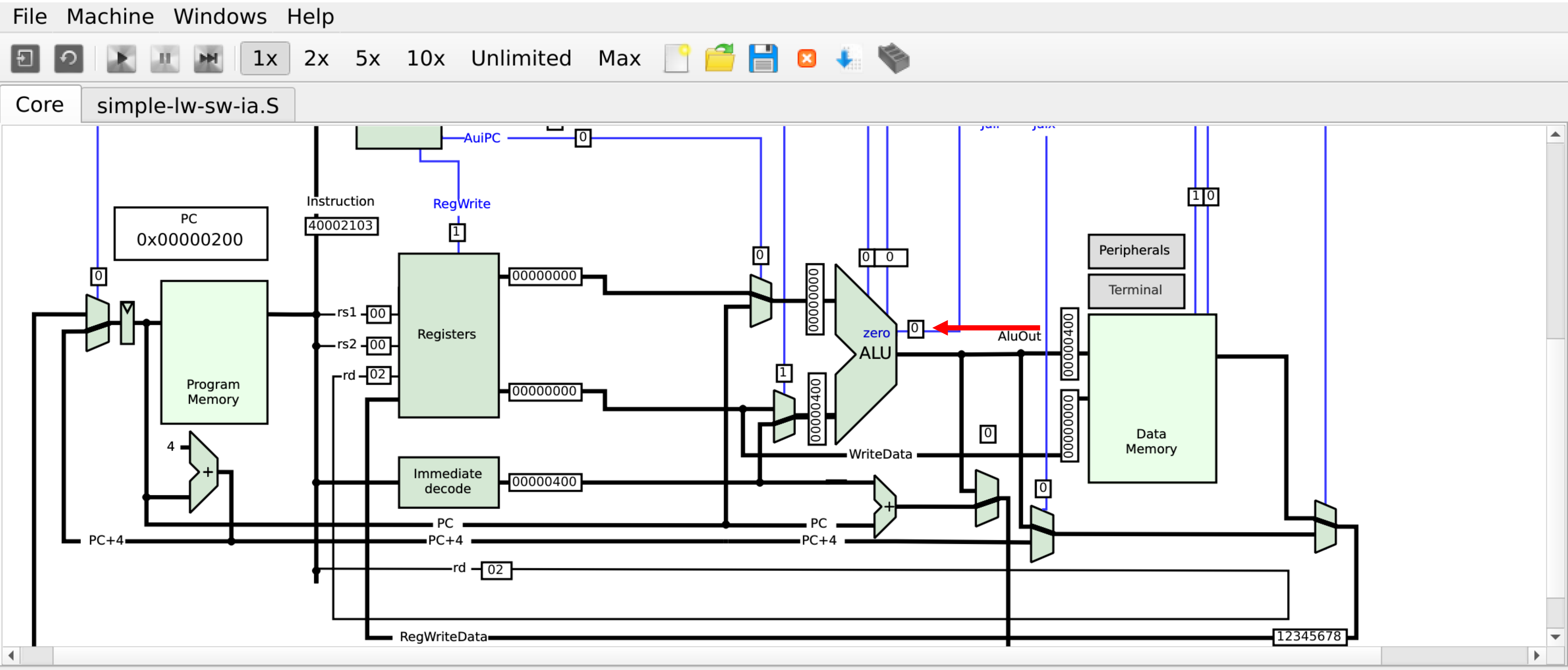
# Core View Detail: Execution



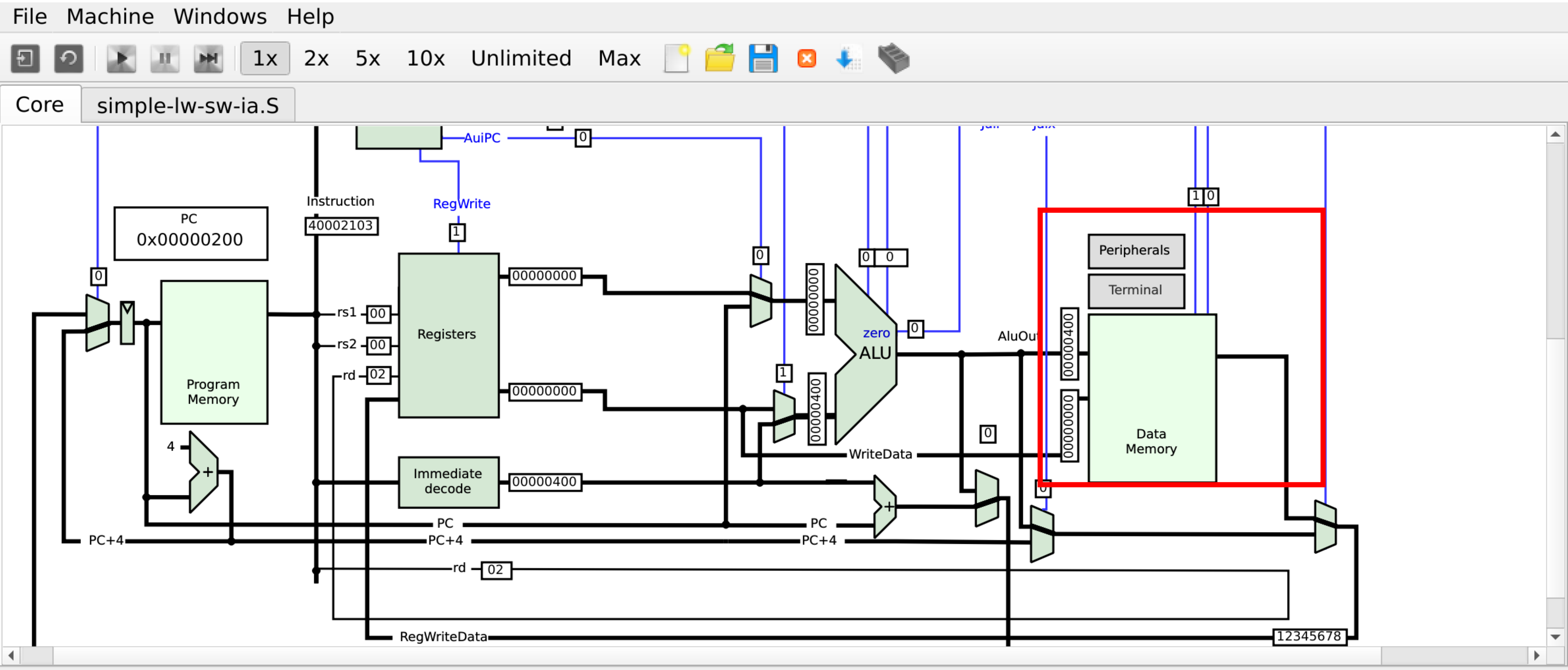
# Core View Detail: Execution



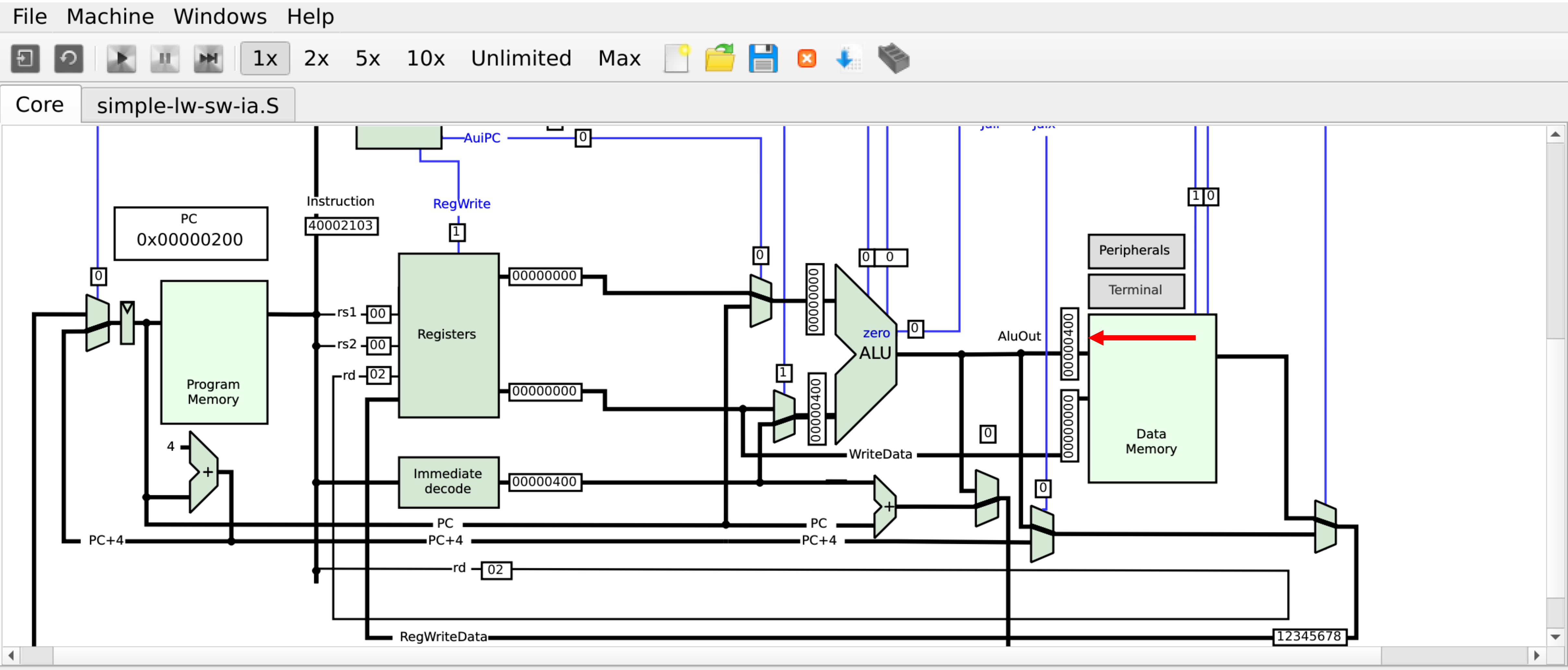
# Core View Detail: Execution



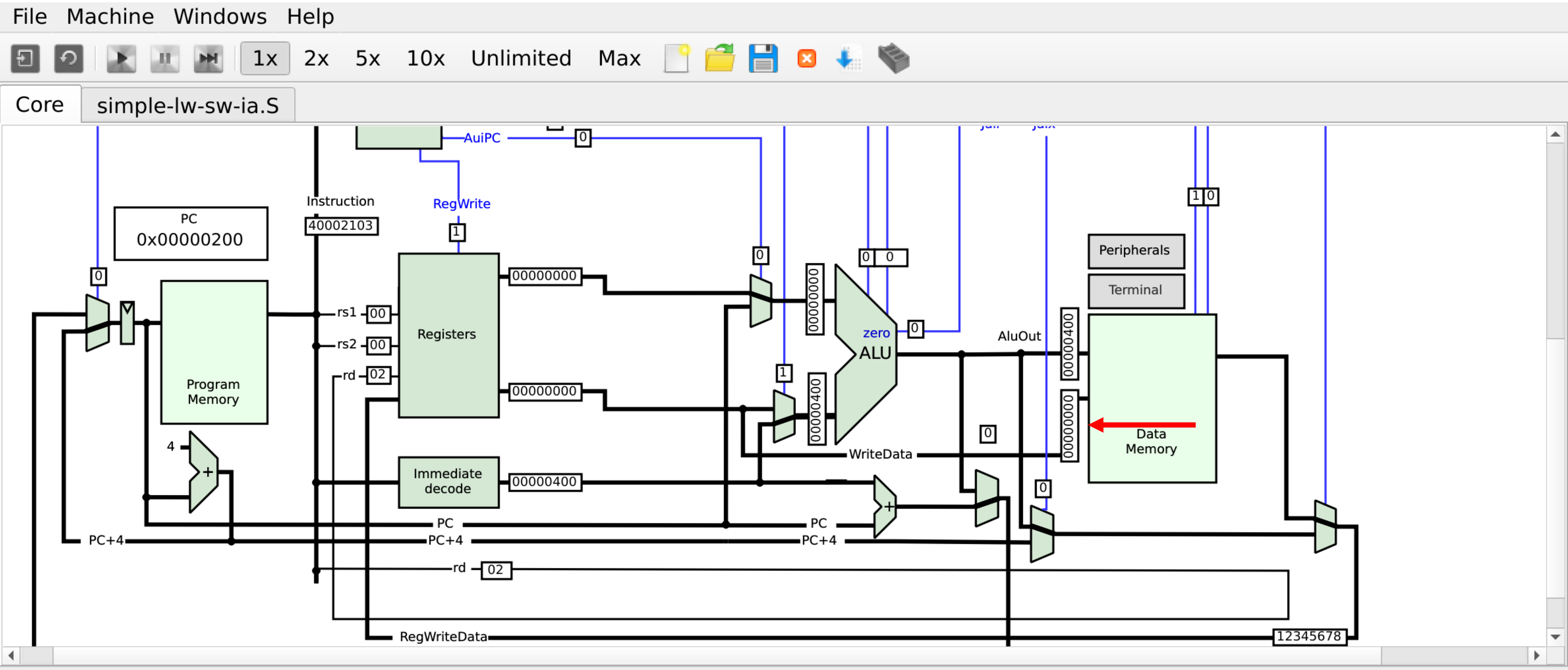
# Core View Detail: Execution



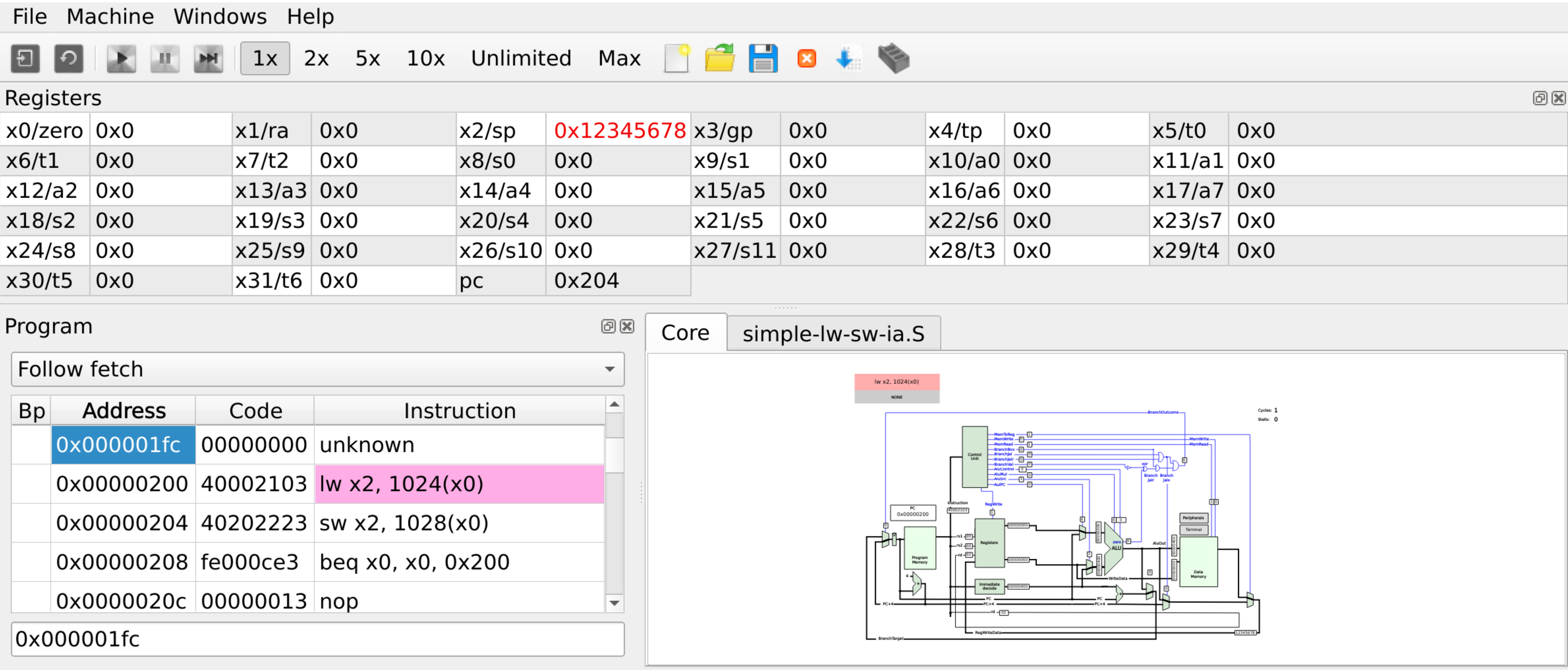
# Core View Detail: Execution



# Core View Detail: Execution



# Load Word

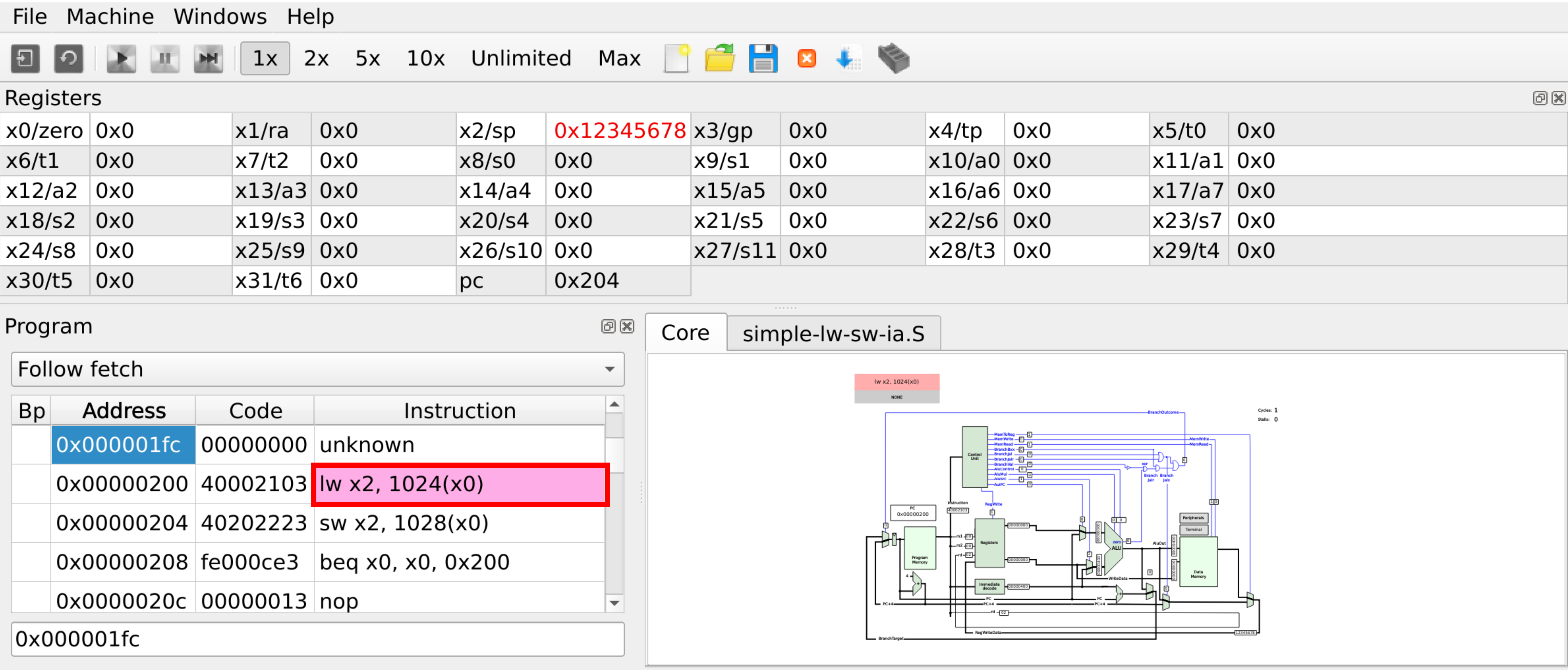


| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| x0/zero | 0x0 | x1/ra | 0x0 | x2/sp | 0x12345678 | x3/gp | 0x0 | x4/tp | 0x0 | x5/t0 | 0x0 |
| x6/t1 | 0x0 | x7/t2 | 0x0 | x8/s0 | 0x0 | x9/s1 | 0x0 | x10/a0 | 0x0 | x11/a1 | 0x0 |
| x12/a2 | 0x0 | x13/a3 | 0x0 | x14/a4 | 0x0 | x15/a5 | 0x0 | x16/a6 | 0x0 | x17/a7 | 0x0 |
| x18/s2 | 0x0 | x19/s3 | 0x0 | x20/s4 | 0x0 | x21/s5 | 0x0 | x22/s6 | 0x0 | x23/s7 | 0x0 |
| x24/s8 | 0x0 | x25/s9 | 0x0 | x26/s10 | 0x0 | x27/s11 | 0x0 | x28/t3 | 0x0 | x29/t4 | 0x0 |
| x30/t5 | 0x0 | x31/t6 | 0x0 | pc | 0x204 | | | | | | |

| Bp | Address | Code | Instruction |
|---|---|---|---|
| | 0x000001fc | 00000000 | unknown |
| | 0x00000200 | 40002103 | lw x2, 1024(x0) |
| | 0x00000204 | 40202223 | sw x2, 1028(x0) |
| | 0x00000208 | fe000ce3 | beq x0, x0, 0x200 |
| | 0x0000020c | 00000013 | nop |

0x000001fc

# Load Word

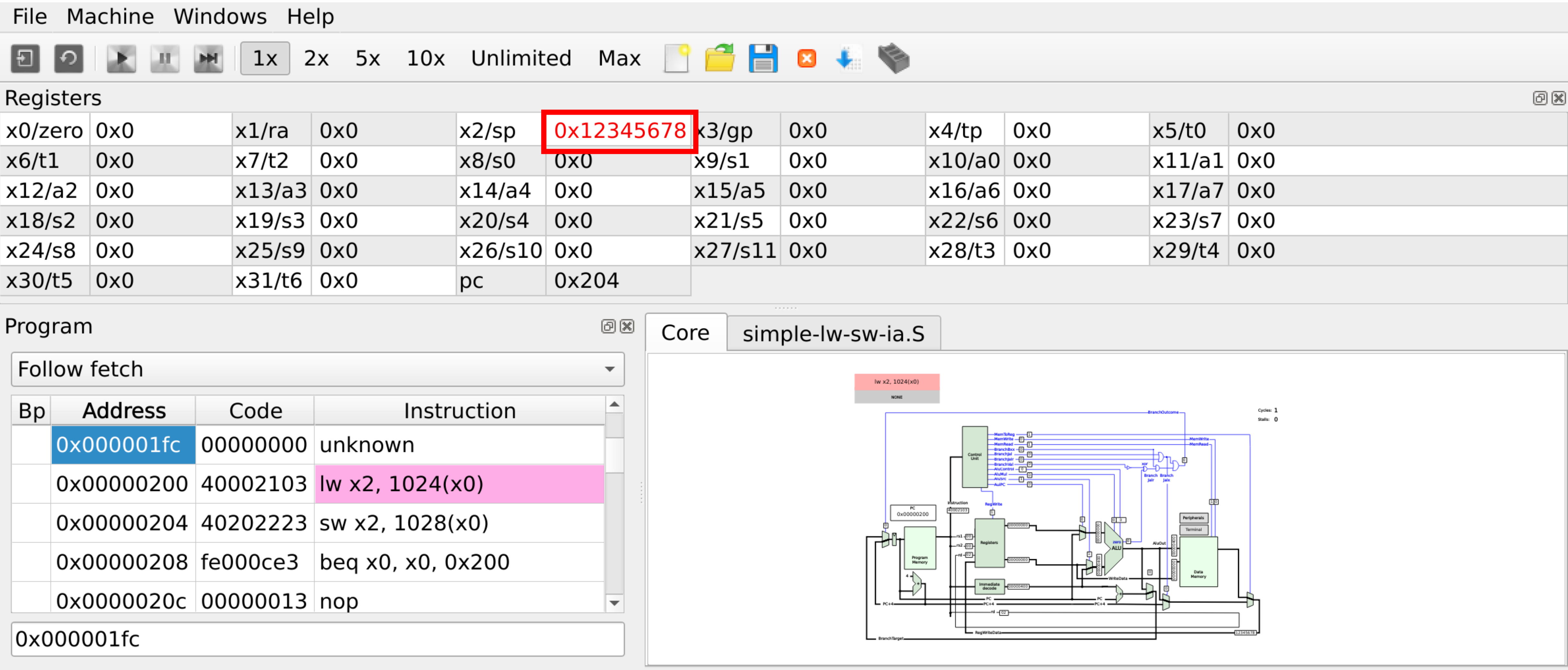


| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| x0/zero | 0x0 | x1/ra | 0x0 | x2/sp | 0x12345678 | x3/gp | 0x0 | x4/tp | 0x0 | x5/t0 | 0x0 |
| x6/t1 | 0x0 | x7/t2 | 0x0 | x8/s0 | 0x0 | x9/s1 | 0x0 | x10/a0 | 0x0 | x11/a1 | 0x0 |
| x12/a2 | 0x0 | x13/a3 | 0x0 | x14/a4 | 0x0 | x15/a5 | 0x0 | x16/a6 | 0x0 | x17/a7 | 0x0 |
| x18/s2 | 0x0 | x19/s3 | 0x0 | x20/s4 | 0x0 | x21/s5 | 0x0 | x22/s6 | 0x0 | x23/s7 | 0x0 |
| x24/s8 | 0x0 | x25/s9 | 0x0 | x26/s10 | 0x0 | x27/s11 | 0x0 | x28/t3 | 0x0 | x29/t4 | 0x0 |
| x30/t5 | 0x0 | x31/t6 | 0x0 | pc | 0x204 | | | | | | |

| Bp | Address | Code | Instruction |
|---|---|---|---|
| | 0x000001fc | 00000000 | unknown |
| | 0x00000200 | 40002103 | lw x2, 1024(x0) |
| | 0x00000204 | 40202223 | sw x2, 1028(x0) |
| | 0x00000208 | fe000ce3 | beq x0, x0, 0x200 |
| | 0x0000020c | 00000013 | nop |

0x000001fc

# Load Word



Registers

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| x0/zero | 0x0 | x1/ra | 0x0 | x2/sp | 0x12345678 | x3/gp | 0x0 | x4/tp | 0x0 | x5/t0 | 0x0 |
| x6/t1 | 0x0 | x7/t2 | 0x0 | x8/s0 | 0x0 | x9/s1 | 0x0 | x10/a0 | 0x0 | x11/a1 | 0x0 |
| x12/a2 | 0x0 | x13/a3 | 0x0 | x14/a4 | 0x0 | x15/a5 | 0x0 | x16/a6 | 0x0 | x17/a7 | 0x0 |
| x18/s2 | 0x0 | x19/s3 | 0x0 | x20/s4 | 0x0 | x21/s5 | 0x0 | x22/s6 | 0x0 | x23/s7 | 0x0 |
| x24/s8 | 0x0 | x25/s9 | 0x0 | x26/s10 | 0x0 | x27/s11 | 0x0 | x28/t3 | 0x0 | x29/t4 | 0x0 |
| x30/t5 | 0x0 | x31/t6 | 0x0 | pc | 0x204 | | | | | | |

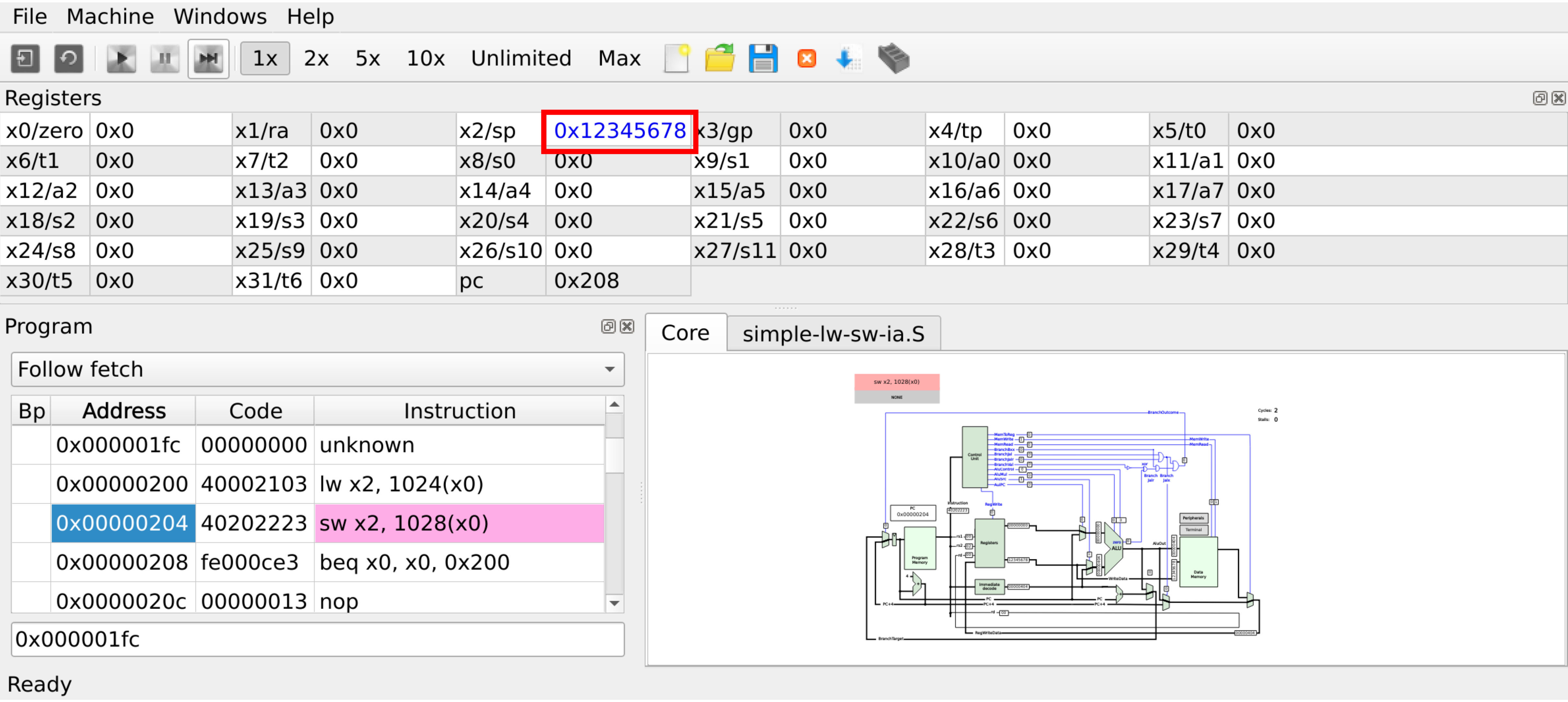
Program

Follow fetch

| Bp | Address | Code | Instruction |
|---|---|---|---|
| | 0x000001fc | 00000000 | unknown |
| | 0x00000200 | 40002103 | lw x2, 1024(x0) |
| | 0x00000204 | 40202223 | sw x2, 1028(x0) |
| | 0x00000208 | fe000ce3 | beq x0, x0, 0x200 |
| | 0x0000020c | 00000013 | nop |

0x000001fc

# Store Word



File Machine Windows Help

1x 2x 5x 10x Unlimited Max

Registers

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| x0/zero | 0x0 | x1/ra | 0x0 | x2/sp | 0x12345678 | x3/gp | 0x0 | x4/tp | 0x0 | x5/t0 | 0x0 |
| x6/t1 | 0x0 | x7/t2 | 0x0 | x8/s0 | 0x0 | x9/s1 | 0x0 | x10/a0 | 0x0 | x11/a1 | 0x0 |
| x12/a2 | 0x0 | x13/a3 | 0x0 | x14/a4 | 0x0 | x15/a5 | 0x0 | x16/a6 | 0x0 | x17/a7 | 0x0 |
| x18/s2 | 0x0 | x19/s3 | 0x0 | x20/s4 | 0x0 | x21/s5 | 0x0 | x22/s6 | 0x0 | x23/s7 | 0x0 |
| x24/s8 | 0x0 | x25/s9 | 0x0 | x26/s10 | 0x0 | x27/s11 | 0x0 | x28/t3 | 0x0 | x29/t4 | 0x0 |
| x30/t5 | 0x0 | x31/t6 | 0x0 | pc | 0x208 | | | | | | |

Program

Follow fetch

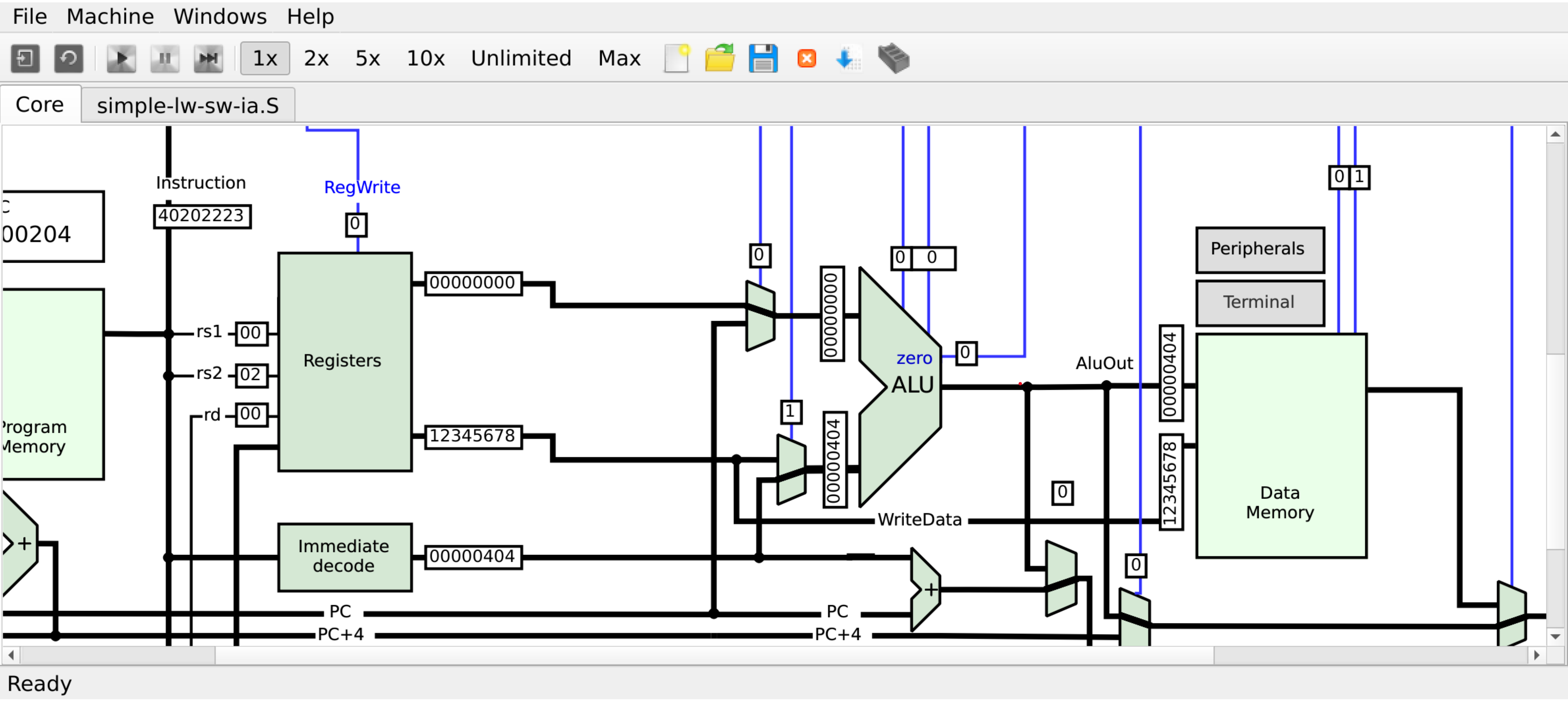
| Bp | Address | Code | Instruction |
|---|---|---|---|
| | 0x000001fc | 00000000 | unknown |
| | 0x00000200 | 40002103 | lw x2, 1024(x0) |
| | 0x00000204 | 40202223 | sw x2, 1028(x0) |
| | 0x00000208 | fe000ce3 | beq x0, x0, 0x200 |
| | 0x0000020c | 00000013 | nop |

0x000001fc

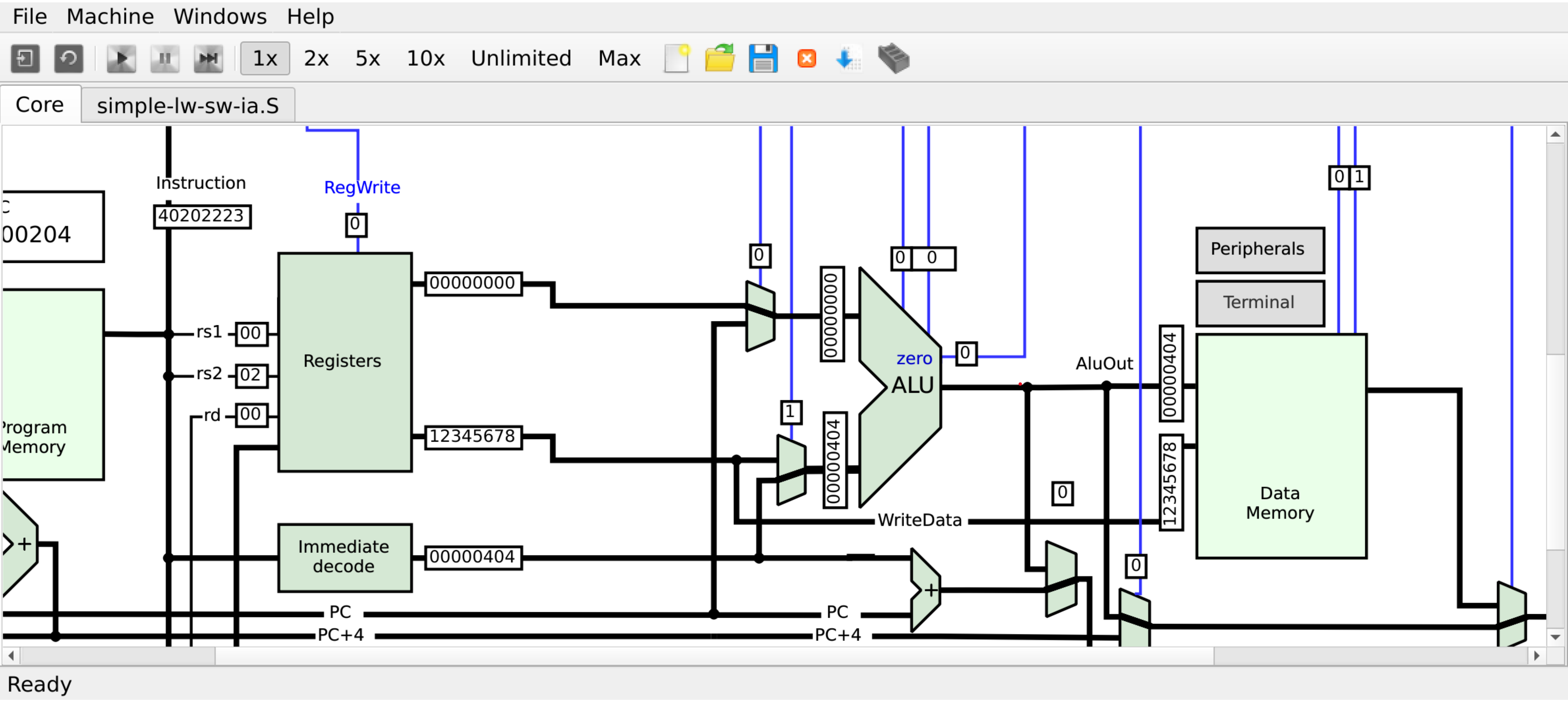
Core simple-lw-sw-ia.S

Ready

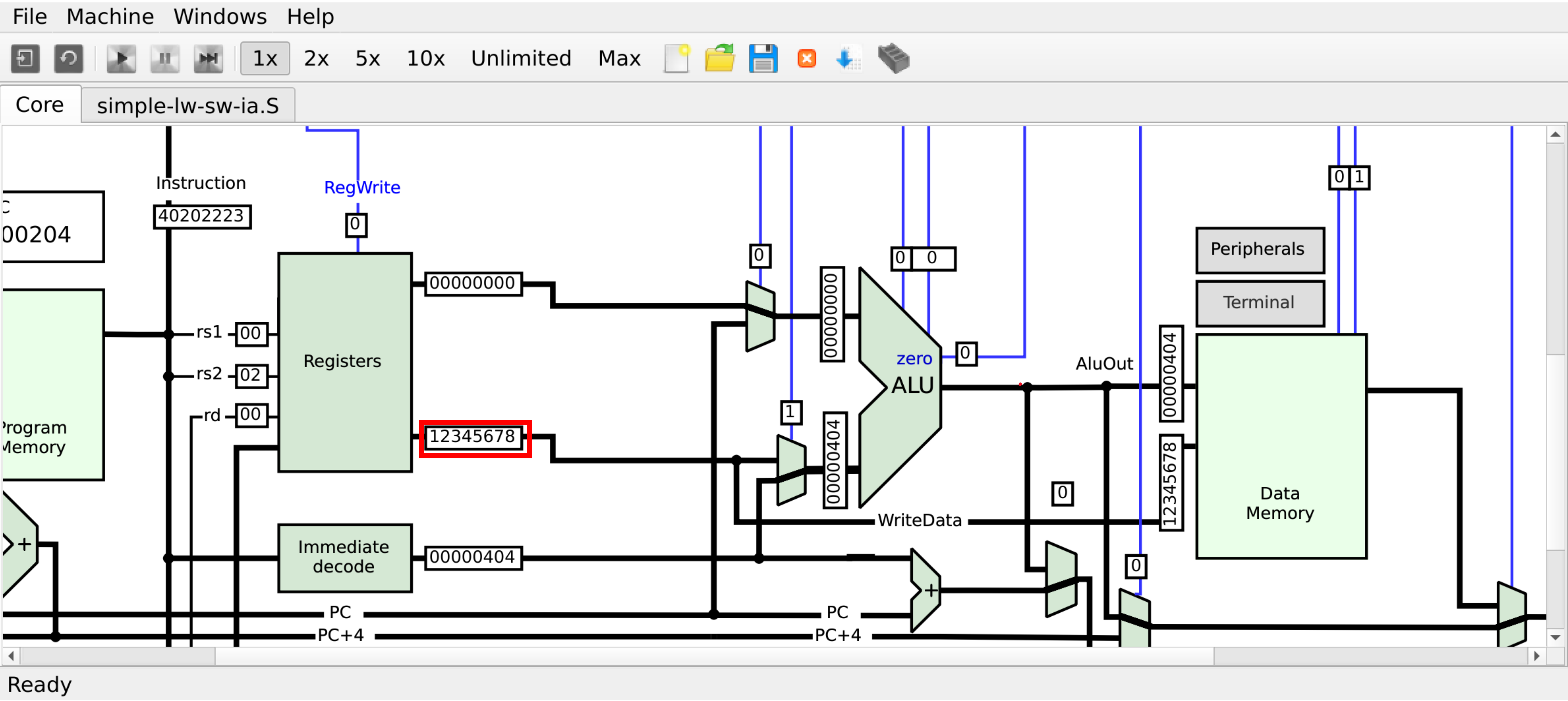
# Store Word Detail



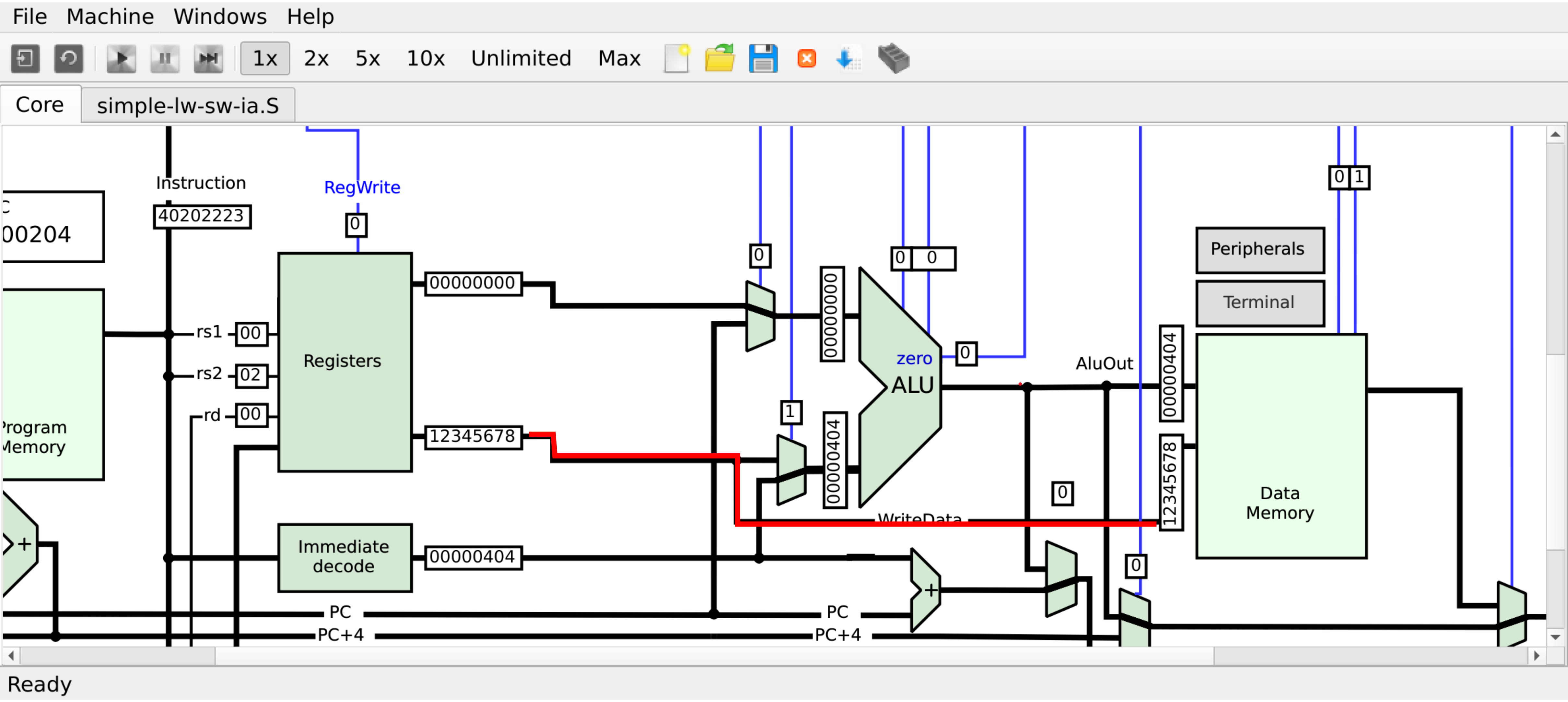
# Store Word Detail



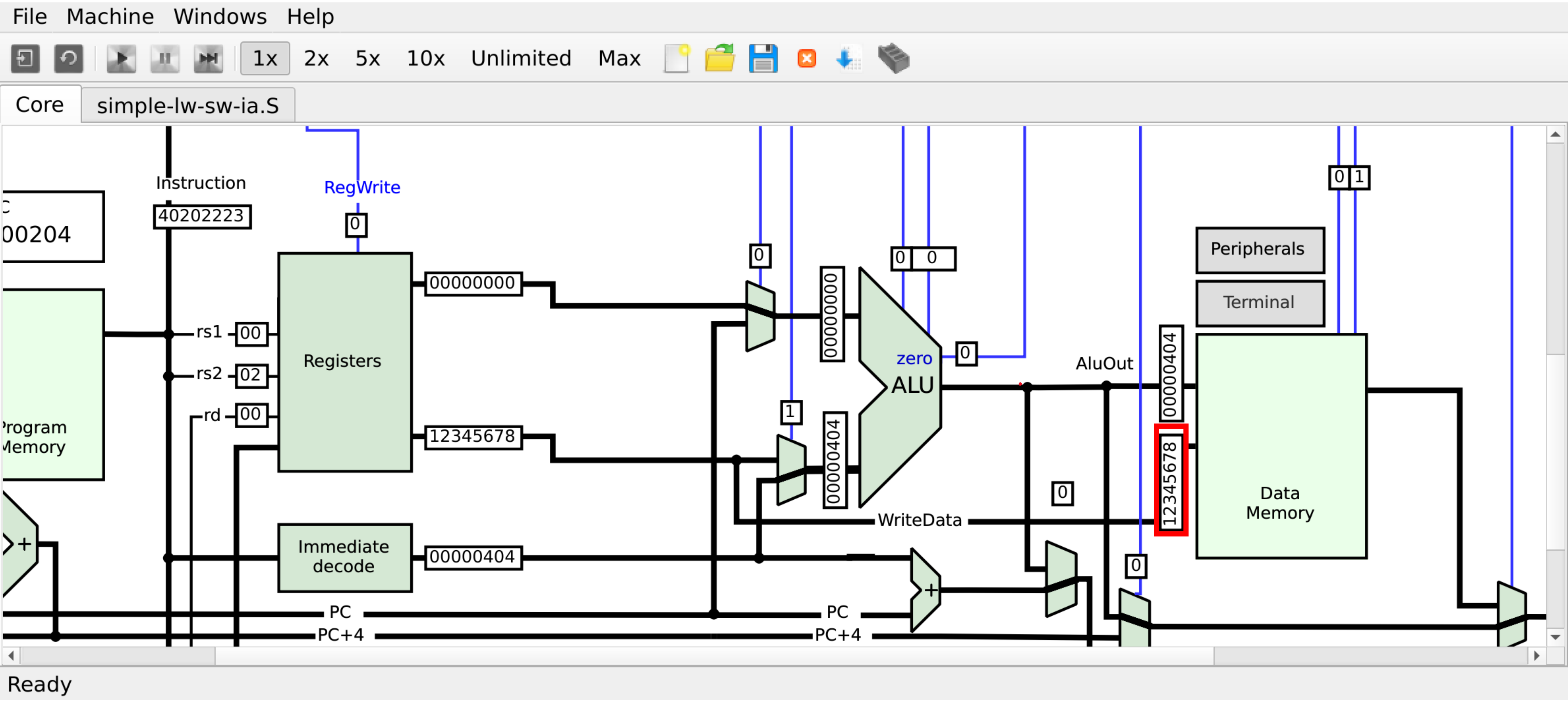
# Store Word Detail



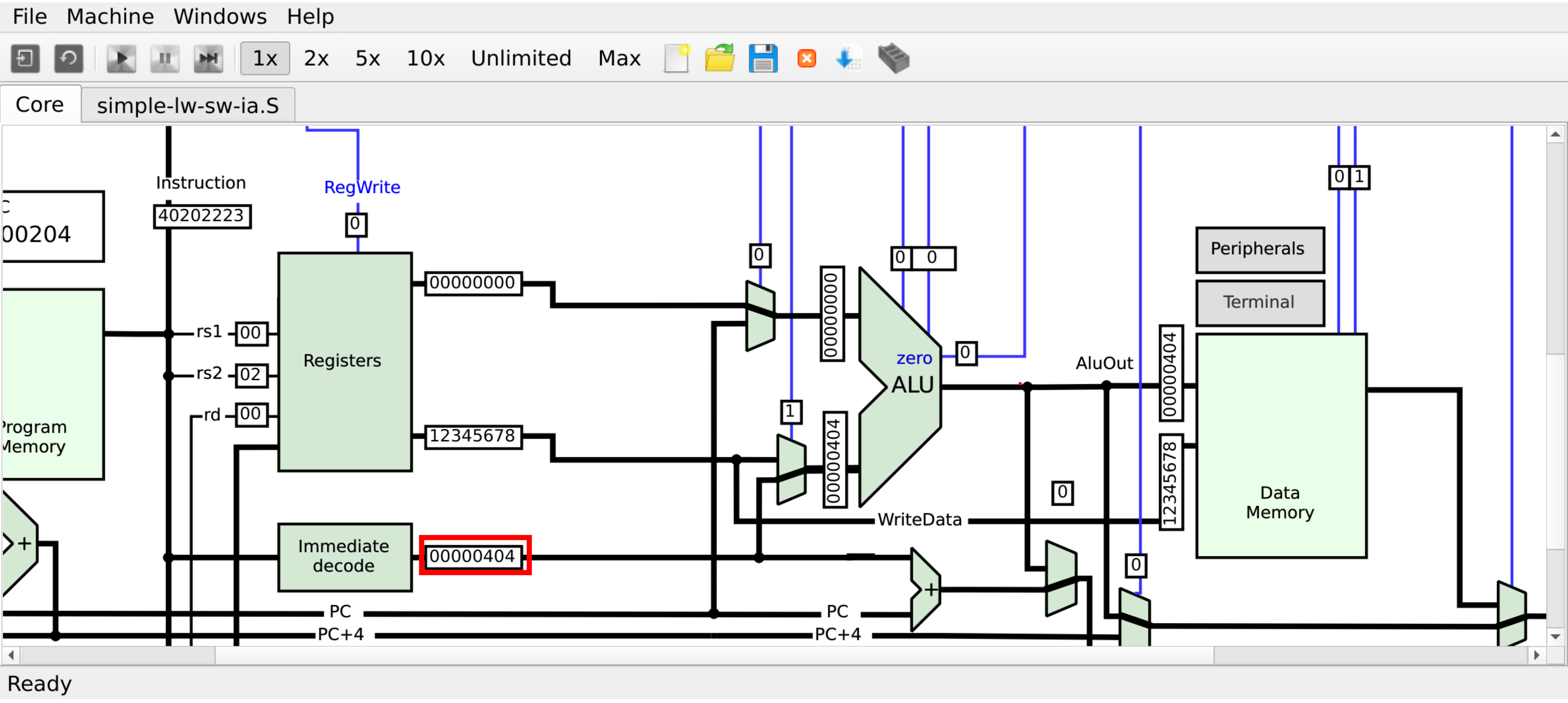
# Store Word Detail



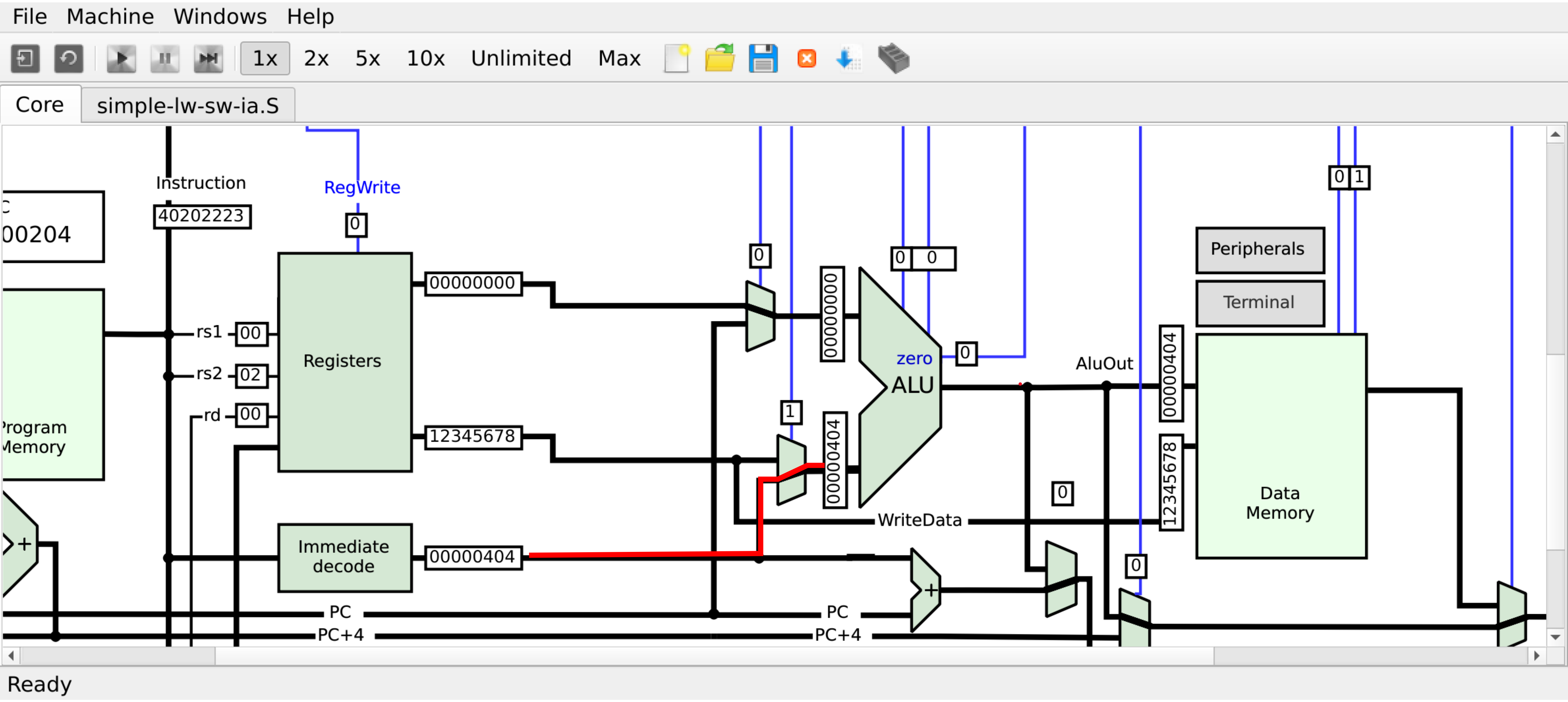
# Store Word Detail



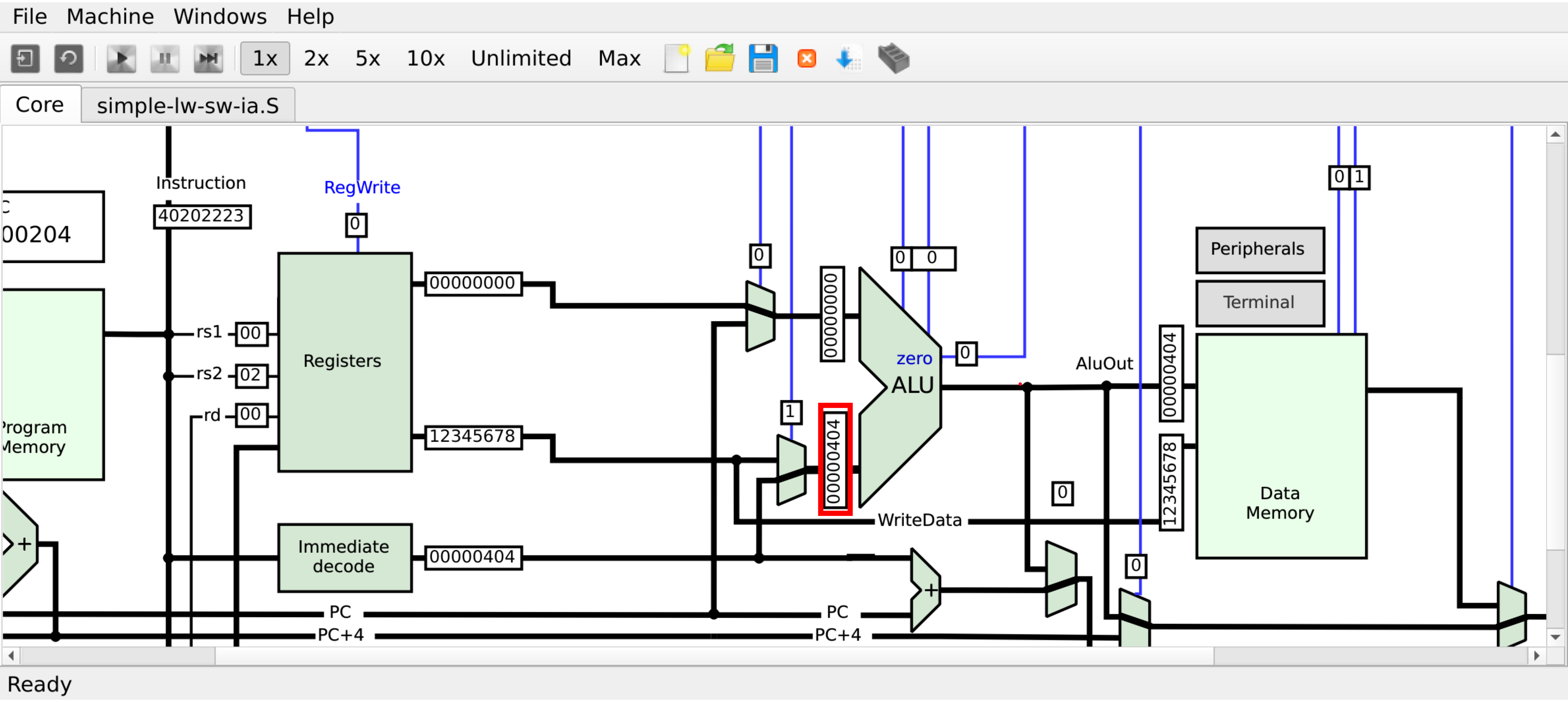
# Store Word Detail



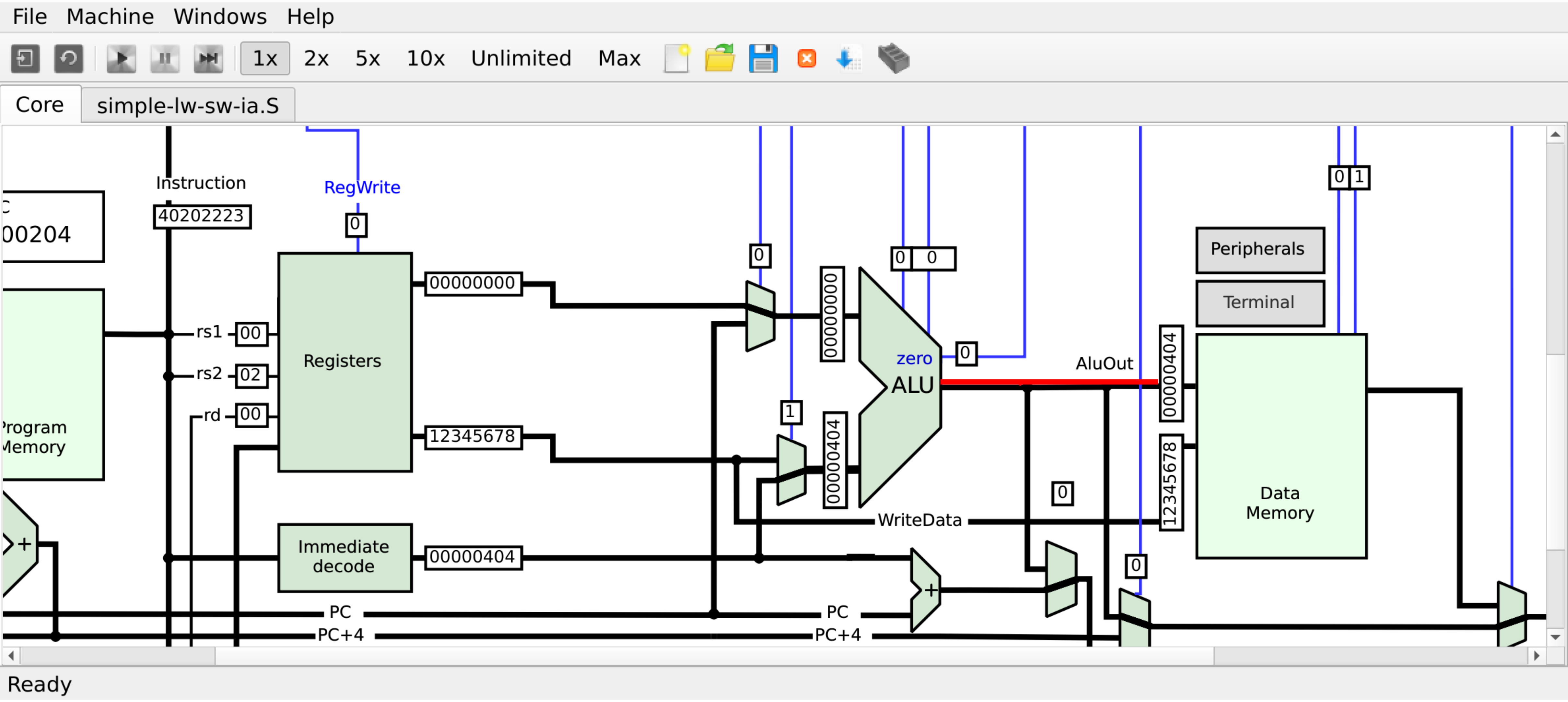
# Store Word Detail



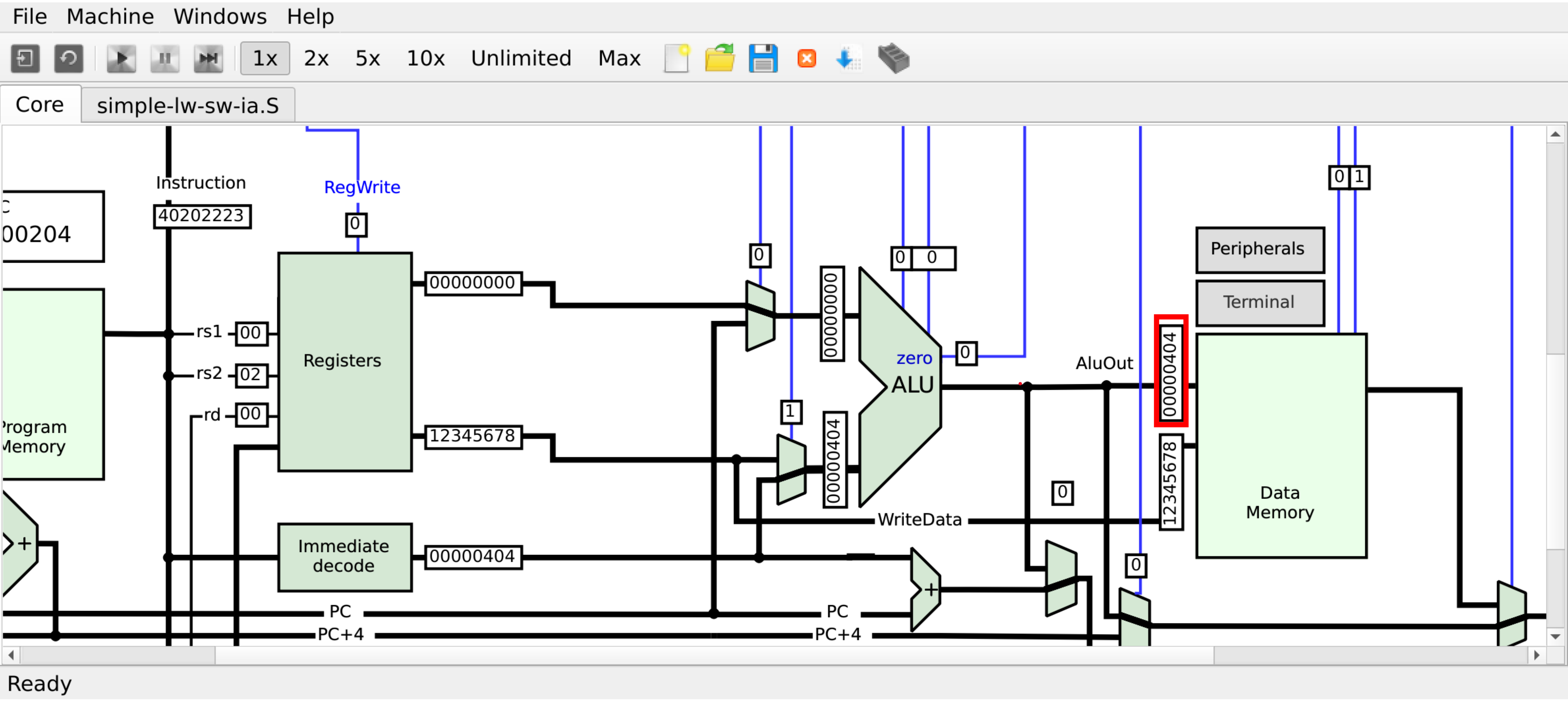
# Store Word Detail



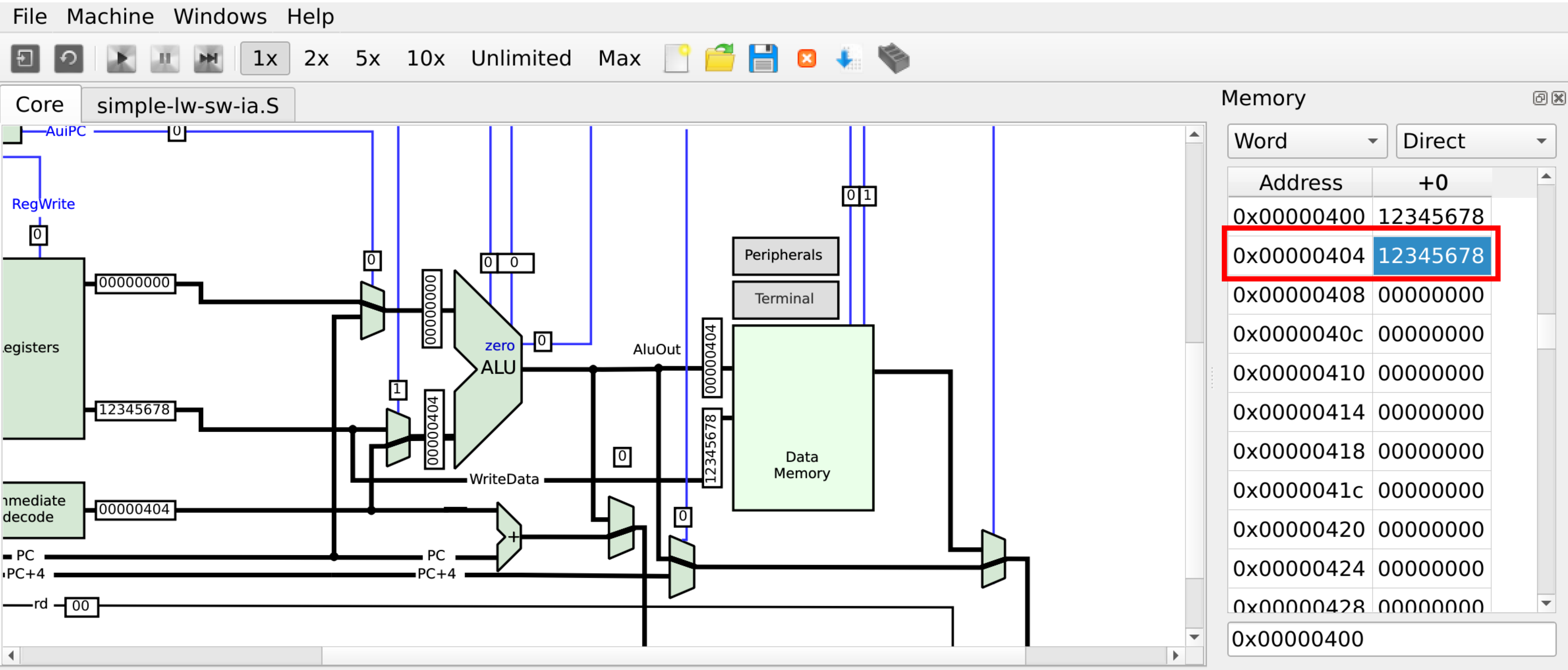
# Store Word Detail



# Store Word Detail



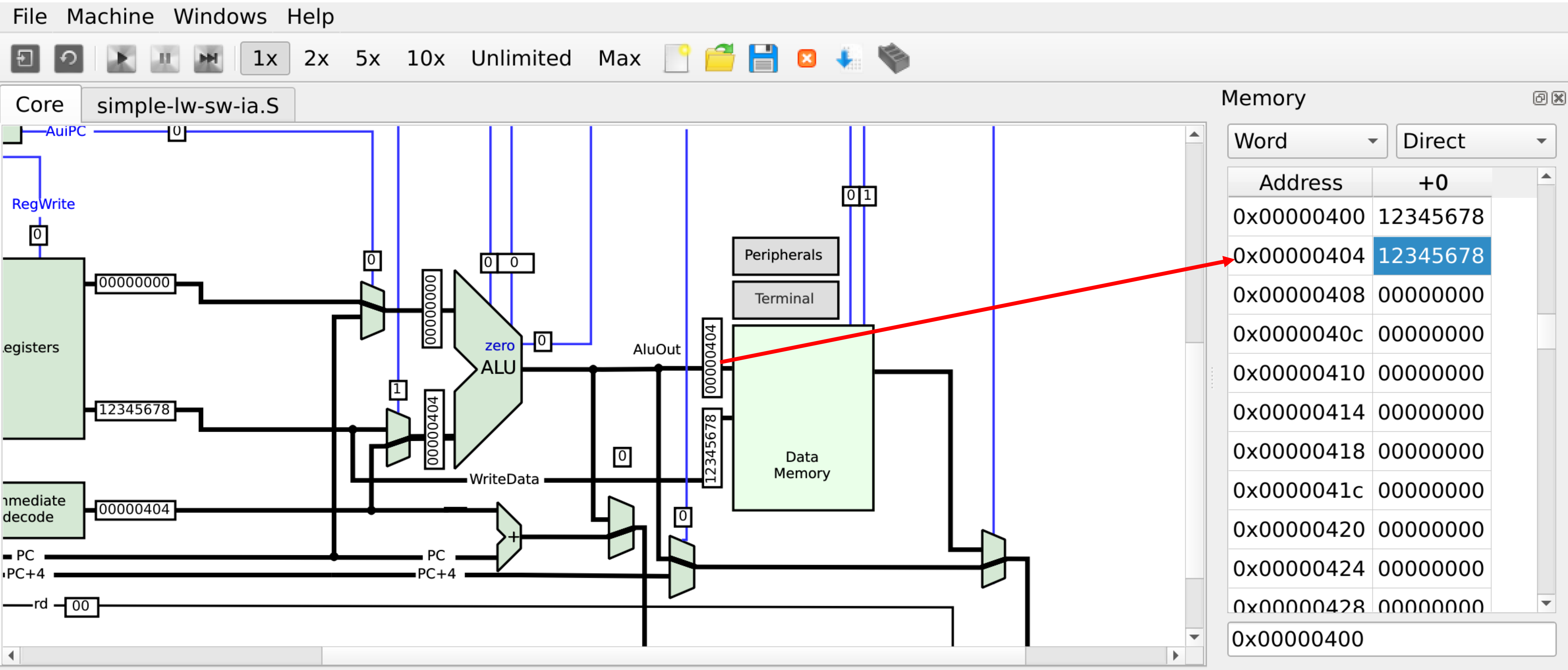
# Memory View



| Address | +0 |
|---|---|
| 0x00000400 | 12345678 |
| 0x00000404 | 12345678 |
| 0x00000408 | 00000000 |
| 0x0000040c | 00000000 |
| 0x00000410 | 00000000 |
| 0x00000414 | 00000000 |
| 0x00000418 | 00000000 |
| 0x0000041c | 00000000 |
| 0x00000420 | 00000000 |
| 0x00000424 | 00000000 |
| 0x00000428 | 00000000 |

0x00000400

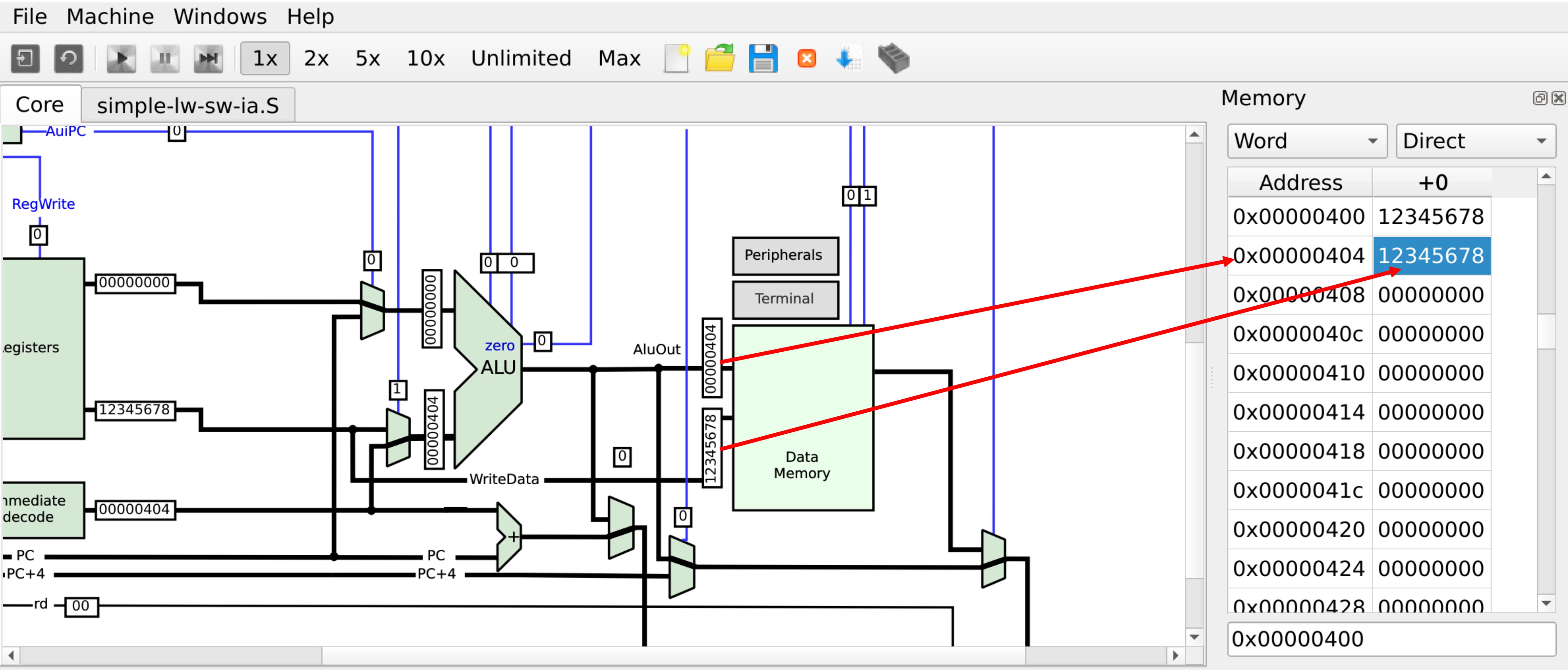
# Memory View



| Address | +0 |
|---|---|
| 0x00000400 | 12345678 |
| 0x00000404 | 12345678 |
| 0x00000408 | 00000000 |
| 0x0000040c | 00000000 |
| 0x00000410 | 00000000 |
| 0x00000414 | 00000000 |
| 0x00000418 | 00000000 |
| 0x0000041c | 00000000 |
| 0x00000420 | 00000000 |
| 0x00000424 | 00000000 |
| 0x00000428 | 00000000 |

0x00000400

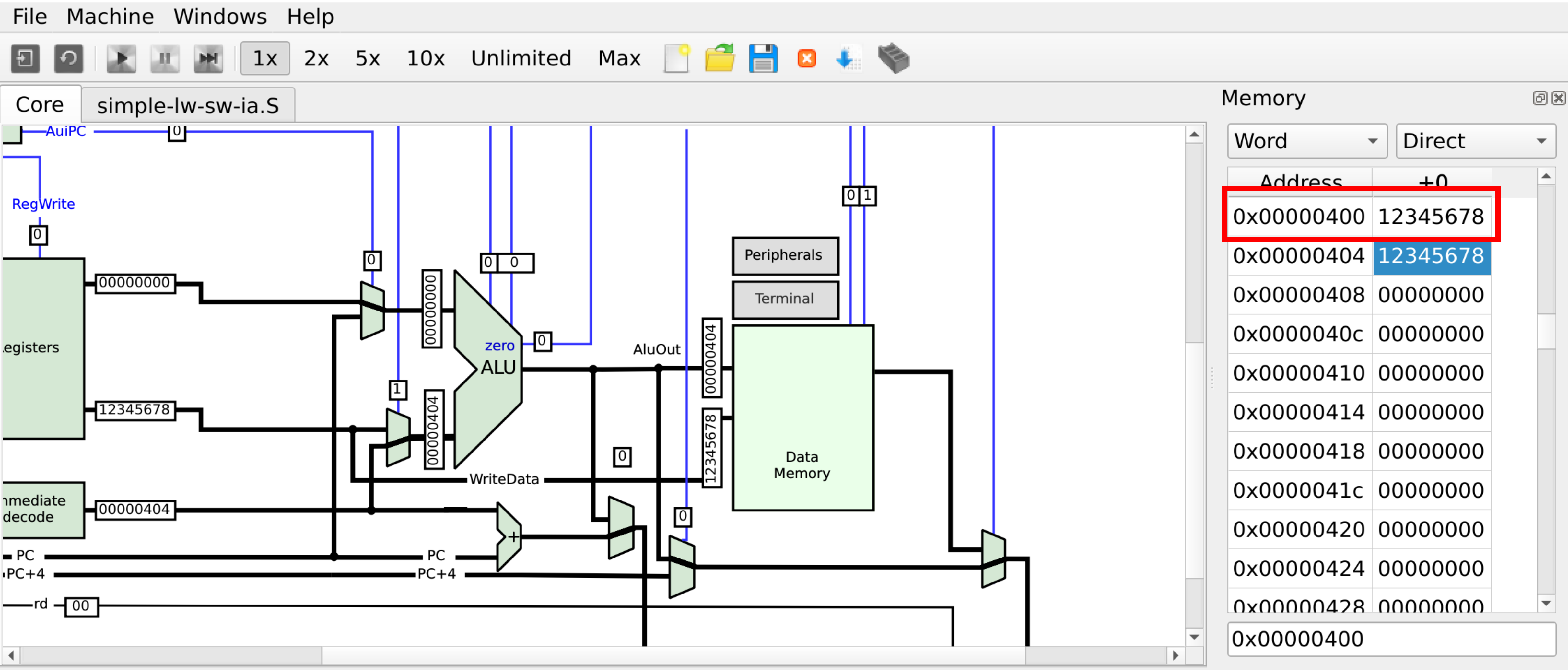
# Memory View



| Address | +0 |
|---|---|
| 0x00000400 | 12345678 |
| 0x00000404 | 12345678 |
| 0x00000408 | 00000000 |
| 0x0000040c | 00000000 |
| 0x00000410 | 00000000 |
| 0x00000414 | 00000000 |
| 0x00000418 | 00000000 |
| 0x0000041c | 00000000 |
| 0x00000420 | 00000000 |
| 0x00000424 | 00000000 |
| 0x00000428 | 00000000 |

0x00000400

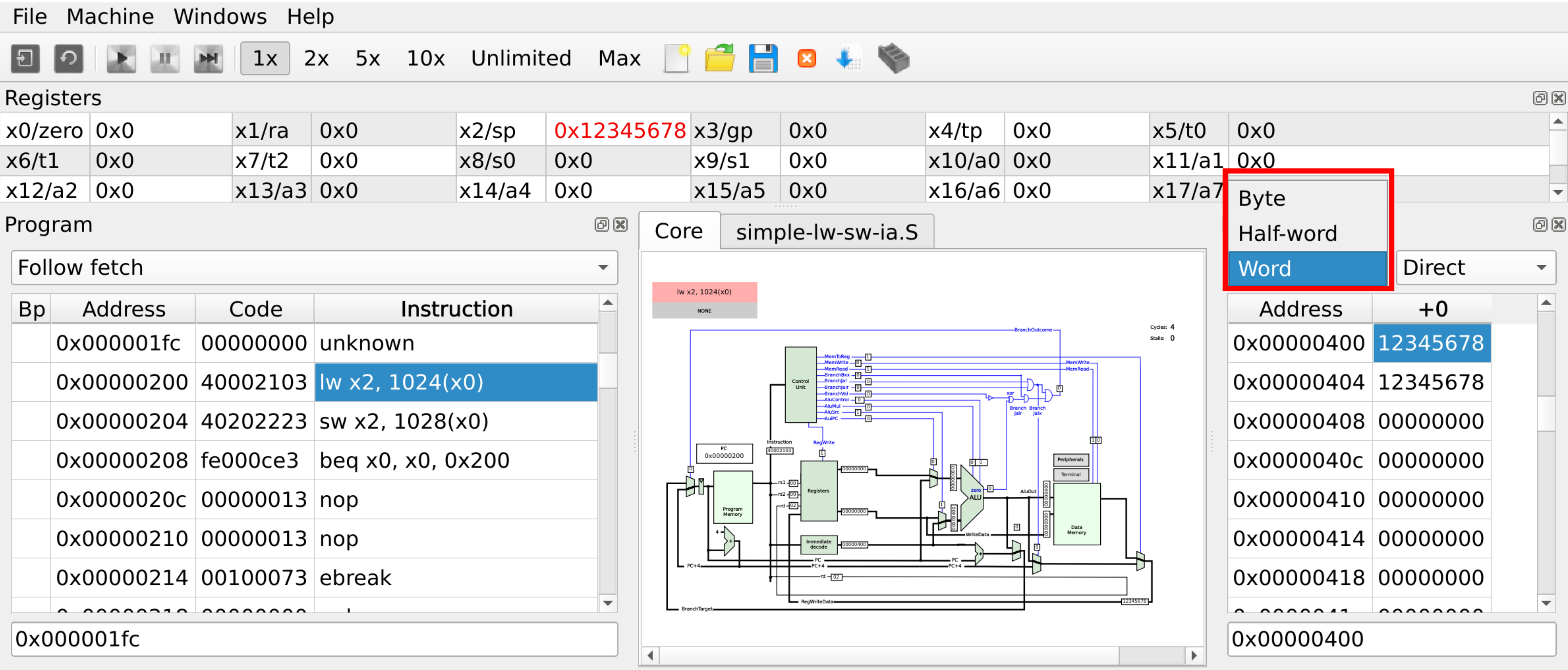
# Memory View



| Address | +0 |
|---|---|
| 0x00000400 | 12345678 |
| 0x00000404 | 12345678 |
| 0x00000408 | 00000000 |
| 0x0000040c | 00000000 |
| 0x00000410 | 00000000 |
| 0x00000414 | 00000000 |
| 0x00000418 | 00000000 |
| 0x0000041c | 00000000 |
| 0x00000420 | 00000000 |
| 0x00000424 | 00000000 |
| 0x00000428 | 00000000 |

0x00000400

# Memory View: Unit Size



| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| x0/zero | 0x0 | x1/ra | 0x0 | x2/sp | 0x12345678 | x3/gp | 0x0 | x4/tp | 0x0 | x5/t0 | 0x0 |
| x6/t1 | 0x0 | x7/t2 | 0x0 | x8/s0 | 0x0 | x9/s1 | 0x0 | x10/a0 | 0x0 | x11/a1 | 0x0 |
| x12/a2 | 0x0 | x13/a3 | 0x0 | x14/a4 | 0x0 | x15/a5 | 0x0 | x16/a6 | 0x0 | x17/a7 | |

Follow fetch

| Bp | Address | Code | Instruction |
|---|---|---|---|
| | 0x000001fc | 00000000 | unknown |
| | 0x00000200 | 40002103 | lw x2, 1024(x0) |
| | 0x00000204 | 40202223 | sw x2, 1028(x0) |
| | 0x00000208 | fe000ce3 | beq x0, x0, 0x200 |
| | 0x0000020c | 00000013 | nop |
| | 0x00000210 | 00000013 | nop |
| | 0x00000214 | 00100073 | ebreak |

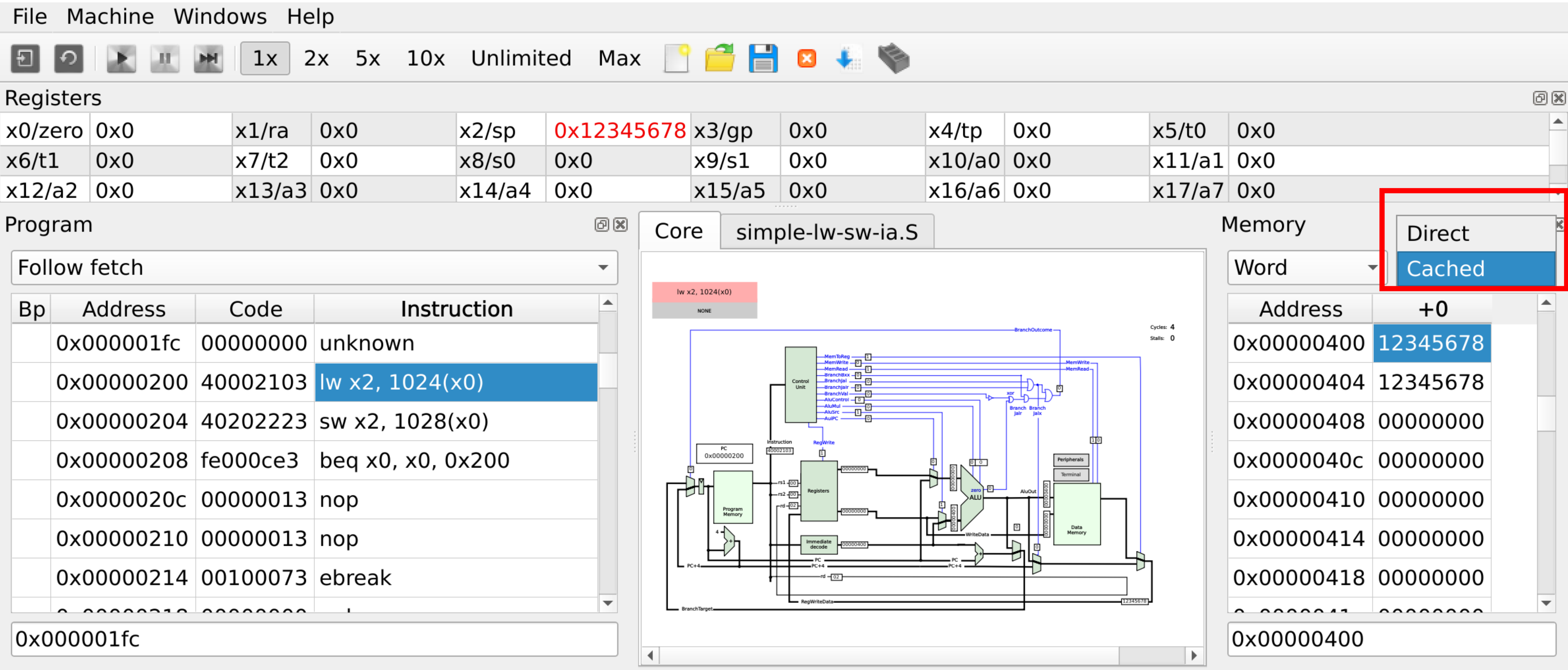
0x000001fc

lw x2, 1024(x0)
NONE
Cycles: 4
Stalls: 0

| Address | +0 |
|---|---|
| 0x00000400 | 12345678 |
| 0x00000404 | 12345678 |
| 0x00000408 | 00000000 |
| 0x0000040c | 00000000 |
| 0x00000410 | 00000000 |
| 0x00000414 | 00000000 |
| 0x00000418 | 00000000 |

0x00000400

# Memory View: Cache


File Machine Windows Help

1x 2x 5x 10x Unlimited Max

Registers

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| x0/zero | 0x0 | x1/ra | 0x0 | x2/sp | 0x12345678 | x3/gp | 0x0 | x4/tp | 0x0 | x5/t0 | 0x0 |
| x6/t1 | 0x0 | x7/t2 | 0x0 | x8/s0 | 0x0 | x9/s1 | 0x0 | x10/a0 | 0x0 | x11/a1 | 0x0 |
| x12/a2 | 0x0 | x13/a3 | 0x0 | x14/a4 | 0x0 | x15/a5 | 0x0 | x16/a6 | 0x0 | x17/a7 | 0x0 |

Program

Follow fetch

| Bp | Address | Code | Instruction |
|---|---|---|---|
| | 0x000001fc | 00000000 | unknown |
| | 0x00000200 | 40002103 | lw x2, 1024(x0) |
| | 0x00000204 | 40202223 | sw x2, 1028(x0) |
| | 0x00000208 | fe000ce3 | beq x0, x0, 0x200 |
| | 0x0000020c | 00000013 | nop |
| | 0x00000210 | 00000013 | nop |
| | 0x00000214 | 00100073 | ebreak |

0x000001fc

Core | simple-lw-sw-ia.S

Memory

Word

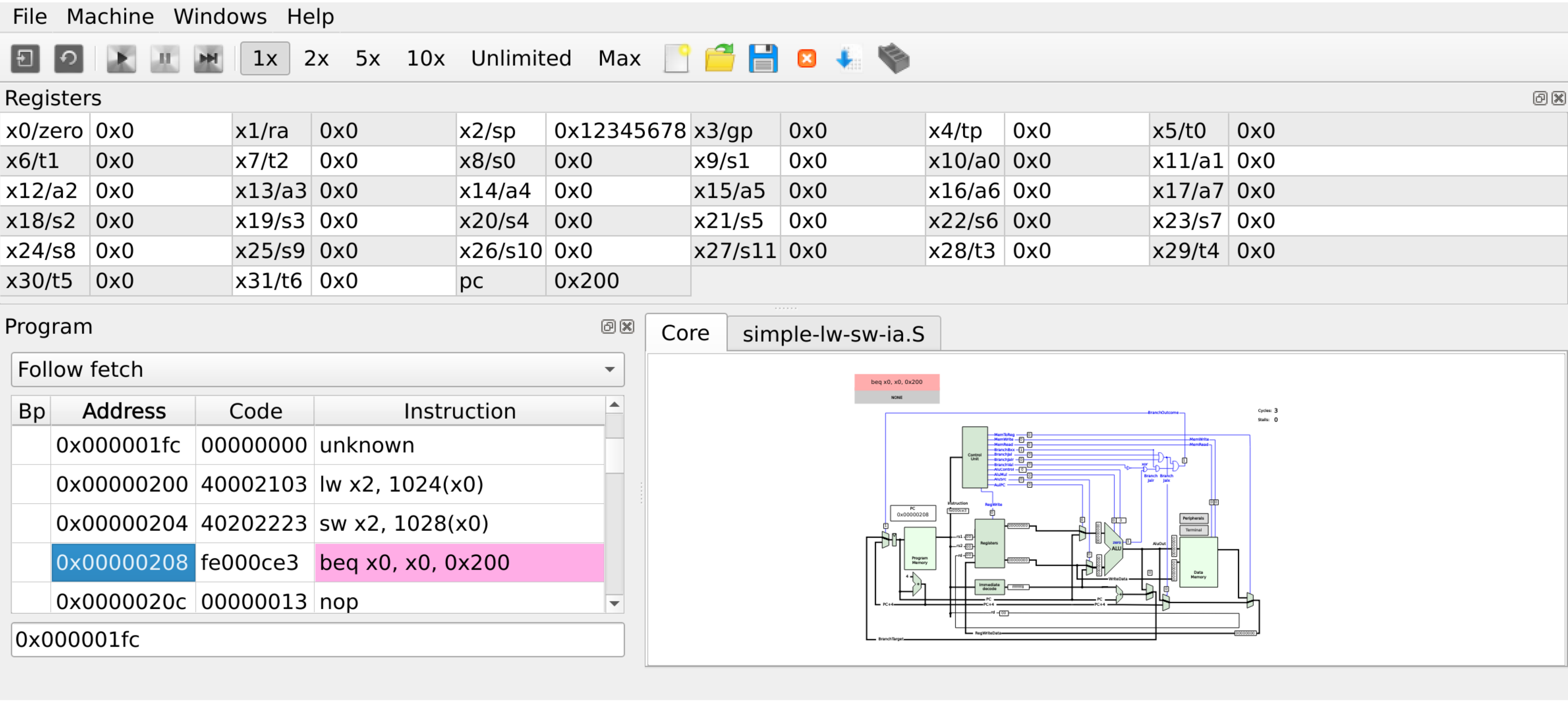
Direct

Cached

| Address | +0 |
|---|---|
| 0x00000400 | 12345678 |
| 0x00000404 | 12345678 |
| 0x00000408 | 00000000 |
| 0x0000040c | 00000000 |
| 0x00000410 | 00000000 |
| 0x00000414 | 00000000 |
| 0x00000418 | 00000000 |

0x00000400

# Branch Equal



Registers

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| x0/zero | 0x0 | x1/ra | 0x0 | x2/sp | 0x12345678 | x3/gp | 0x0 | x4/tp | 0x0 | x5/t0 | 0x0 |
| x6/t1 | 0x0 | x7/t2 | 0x0 | x8/s0 | 0x0 | x9/s1 | 0x0 | x10/a0 | 0x0 | x11/a1 | 0x0 |
| x12/a2 | 0x0 | x13/a3 | 0x0 | x14/a4 | 0x0 | x15/a5 | 0x0 | x16/a6 | 0x0 | x17/a7 | 0x0 |
| x18/s2 | 0x0 | x19/s3 | 0x0 | x20/s4 | 0x0 | x21/s5 | 0x0 | x22/s6 | 0x0 | x23/s7 | 0x0 |
| x24/s8 | 0x0 | x25/s9 | 0x0 | x26/s10 | 0x0 | x27/s11 | 0x0 | x28/t3 | 0x0 | x29/t4 | 0x0 |
| x30/t5 | 0x0 | x31/t6 | 0x0 | pc | 0x200 | | | | | | |

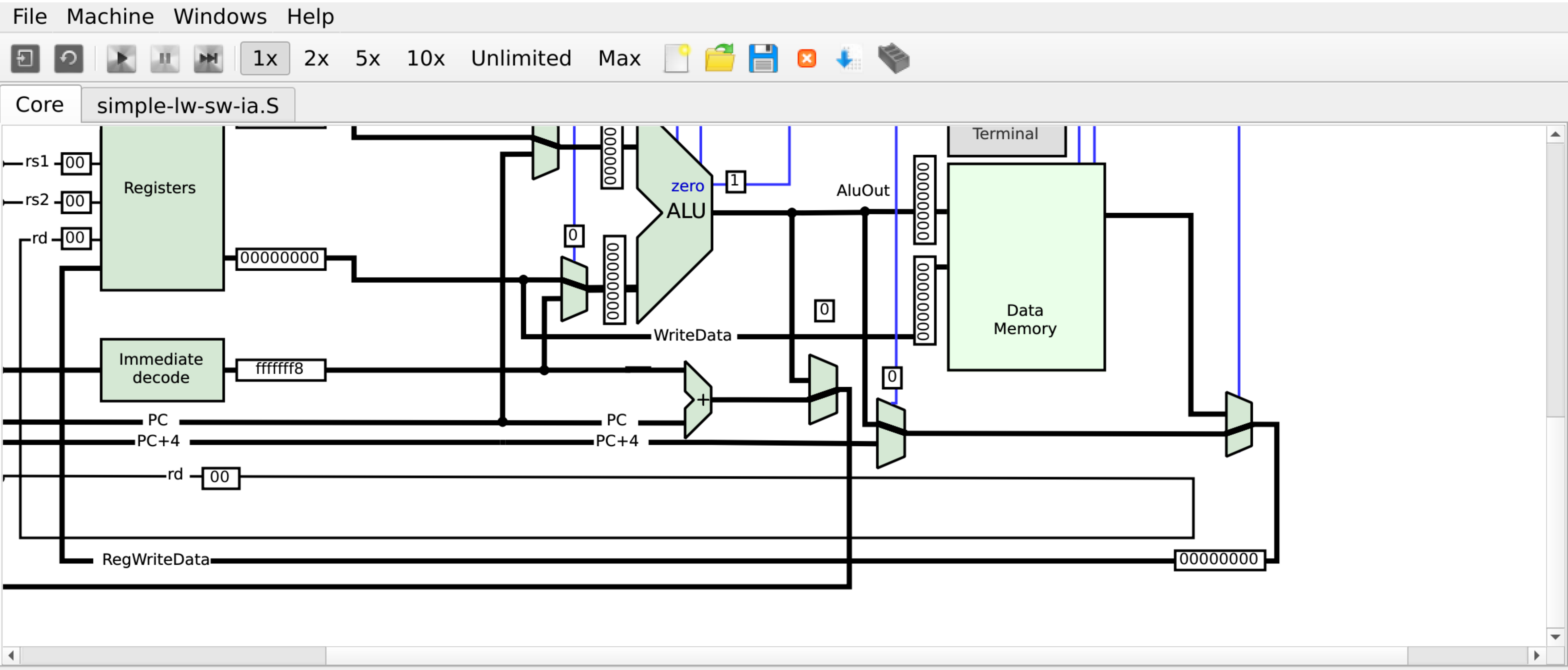
Program

Follow fetch

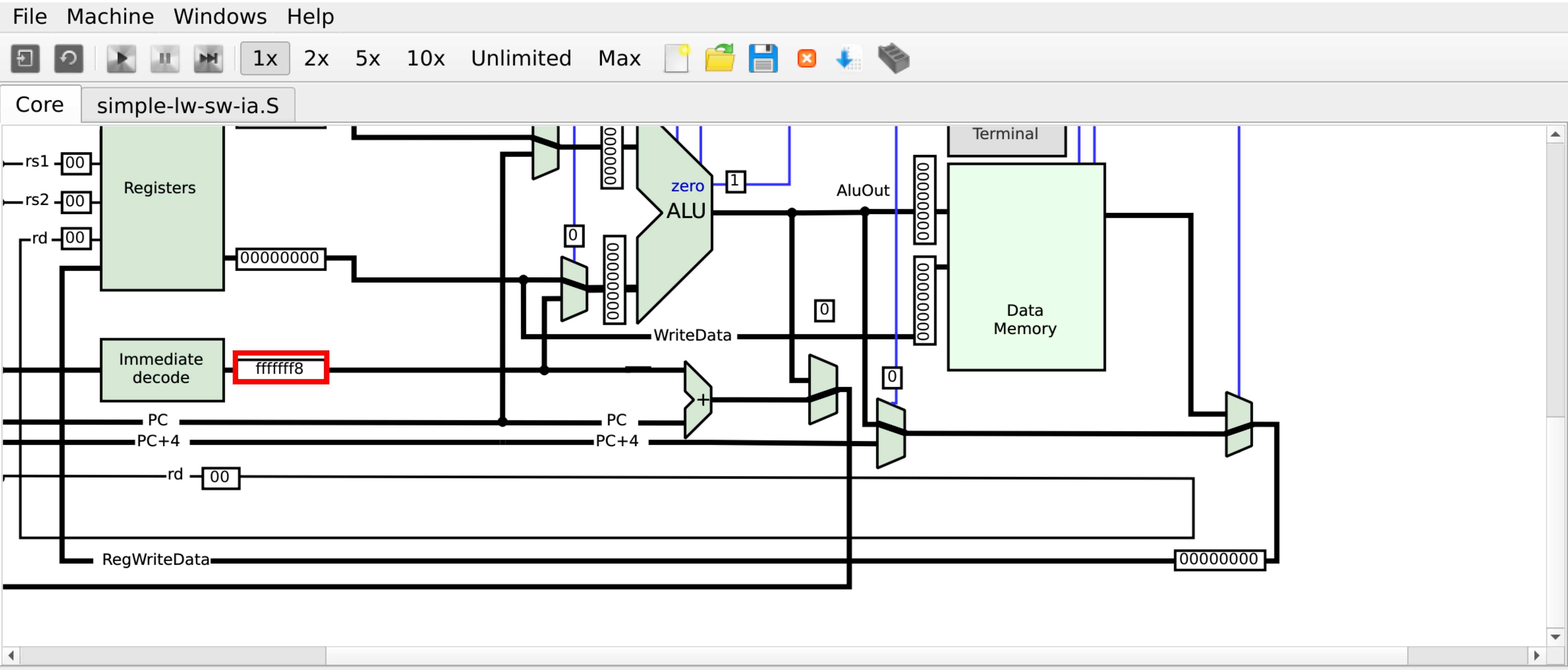
| Bp | Address | Code | Instruction |
|---|---|---|---|
| | 0x000001fc | 00000000 | unknown |
| | 0x00000200 | 40002103 | lw x2, 1024(x0) |
| | 0x00000204 | 40202223 | sw x2, 1028(x0) |
| | 0x00000208 | fe000ce3 | beq x0, x0, 0x200 |
| | 0x0000020c | 00000013 | nop |

0x000001fc

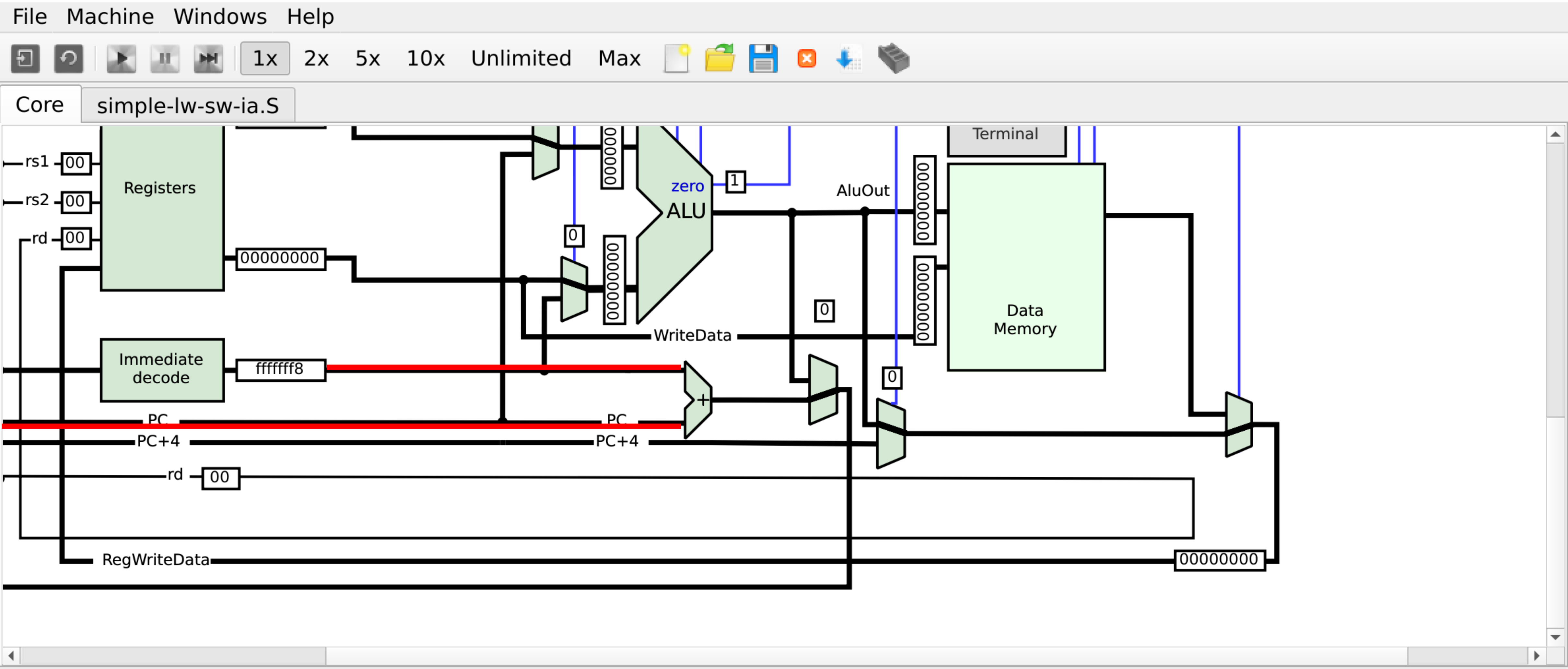
# Branch Target



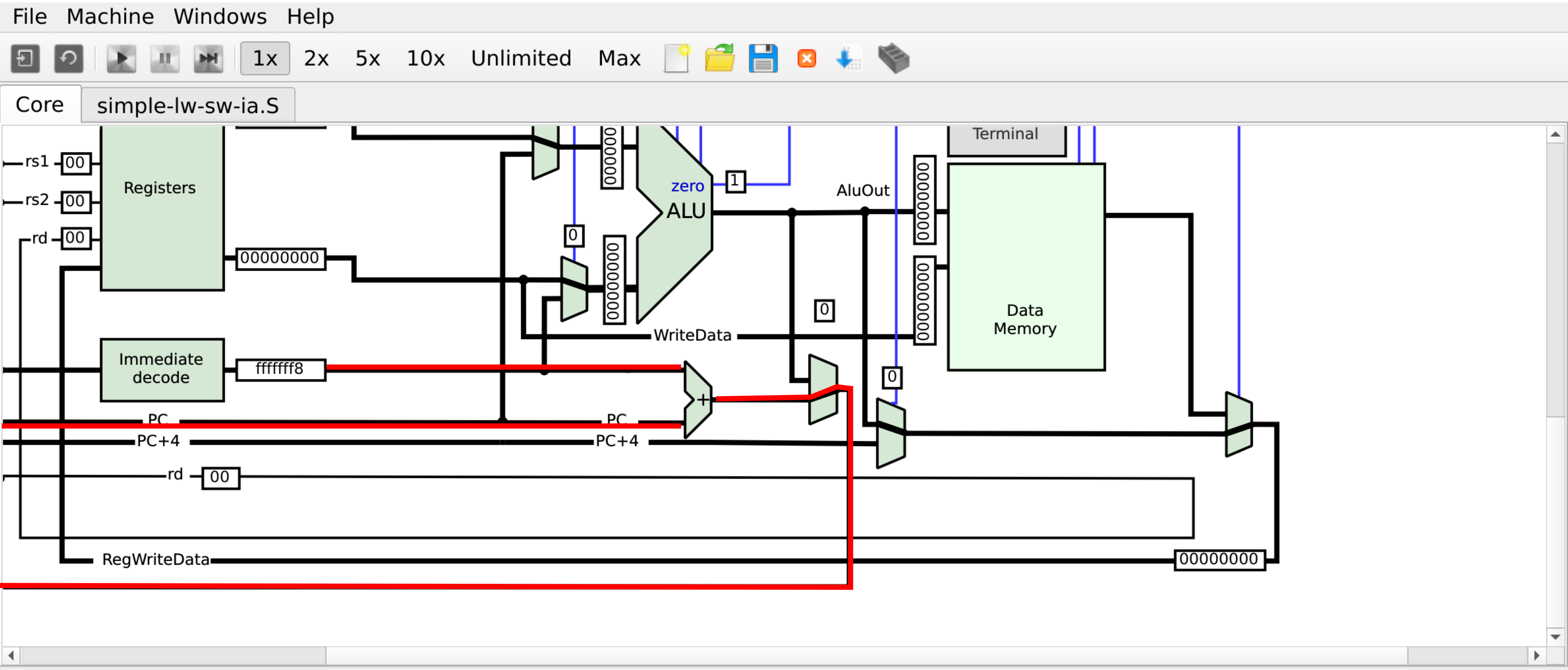
# Branch Target



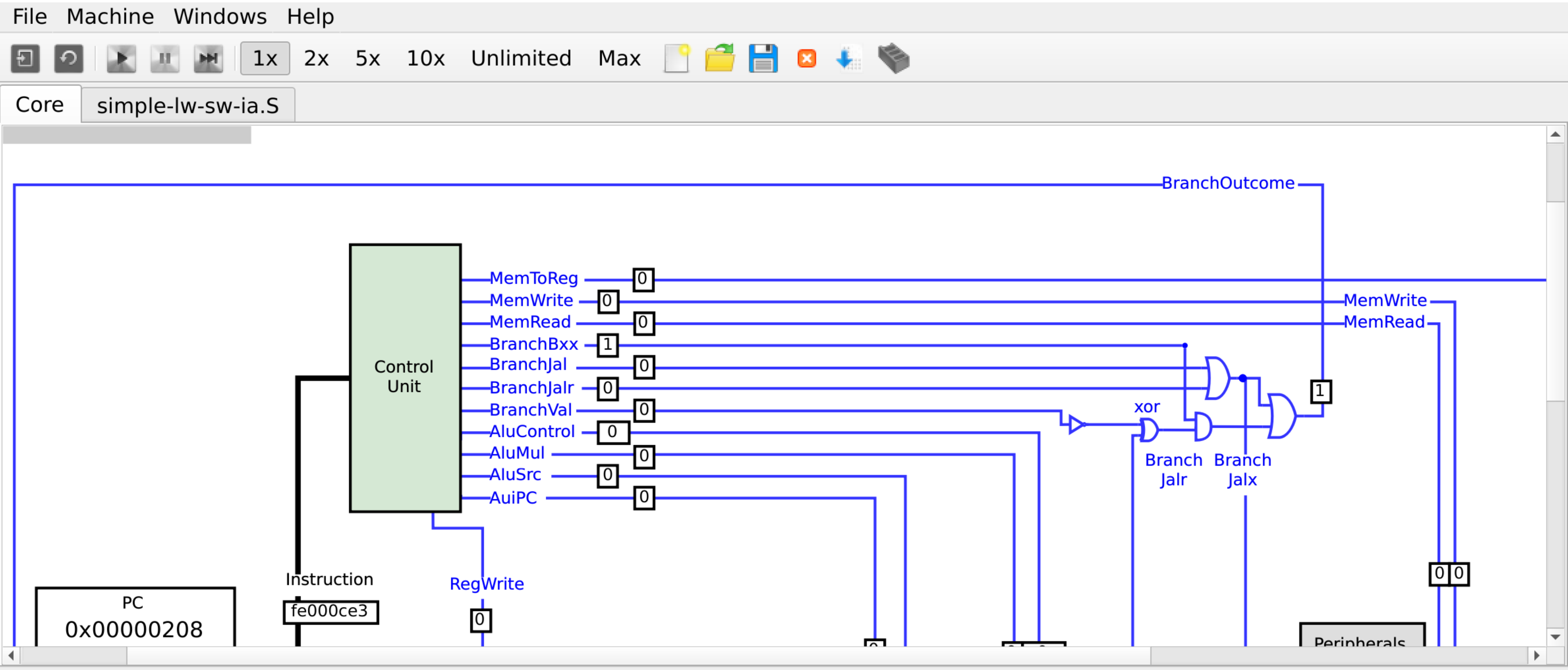
# Branch Target



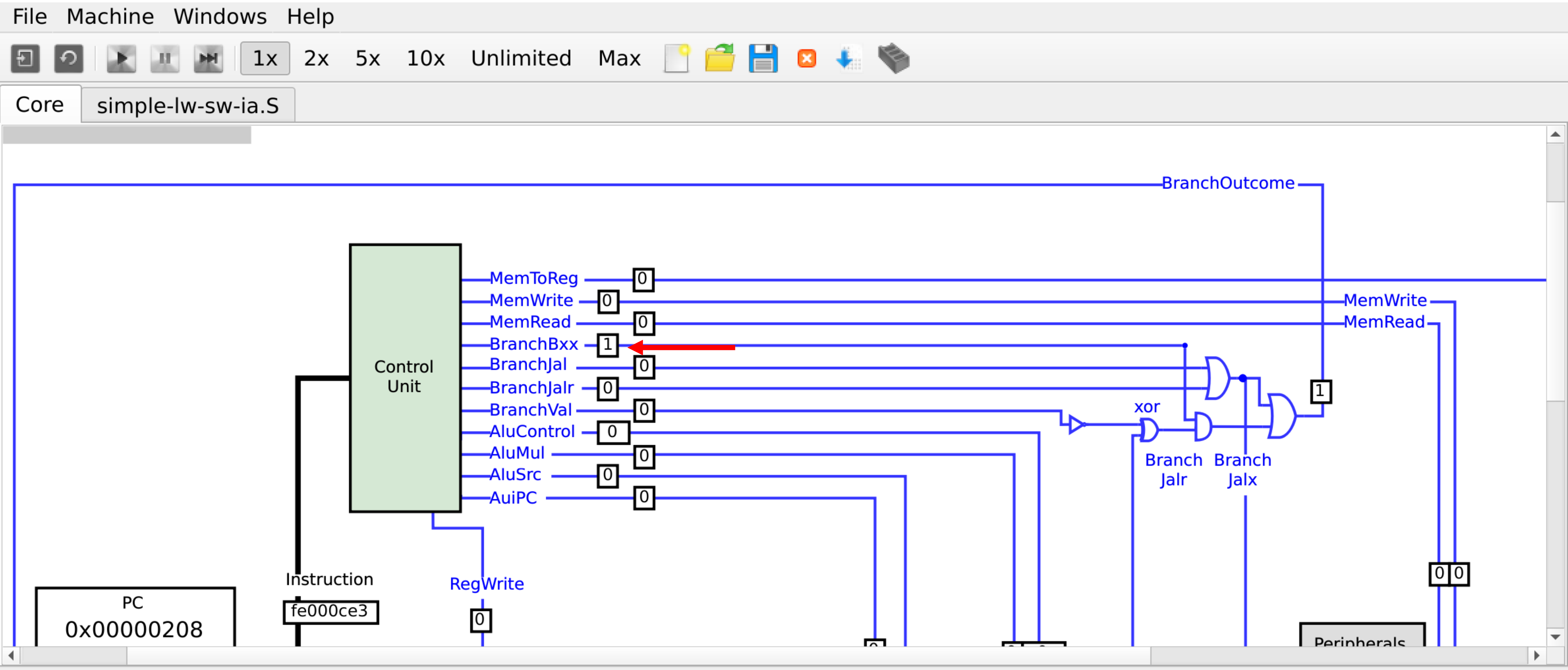
# Branch Target



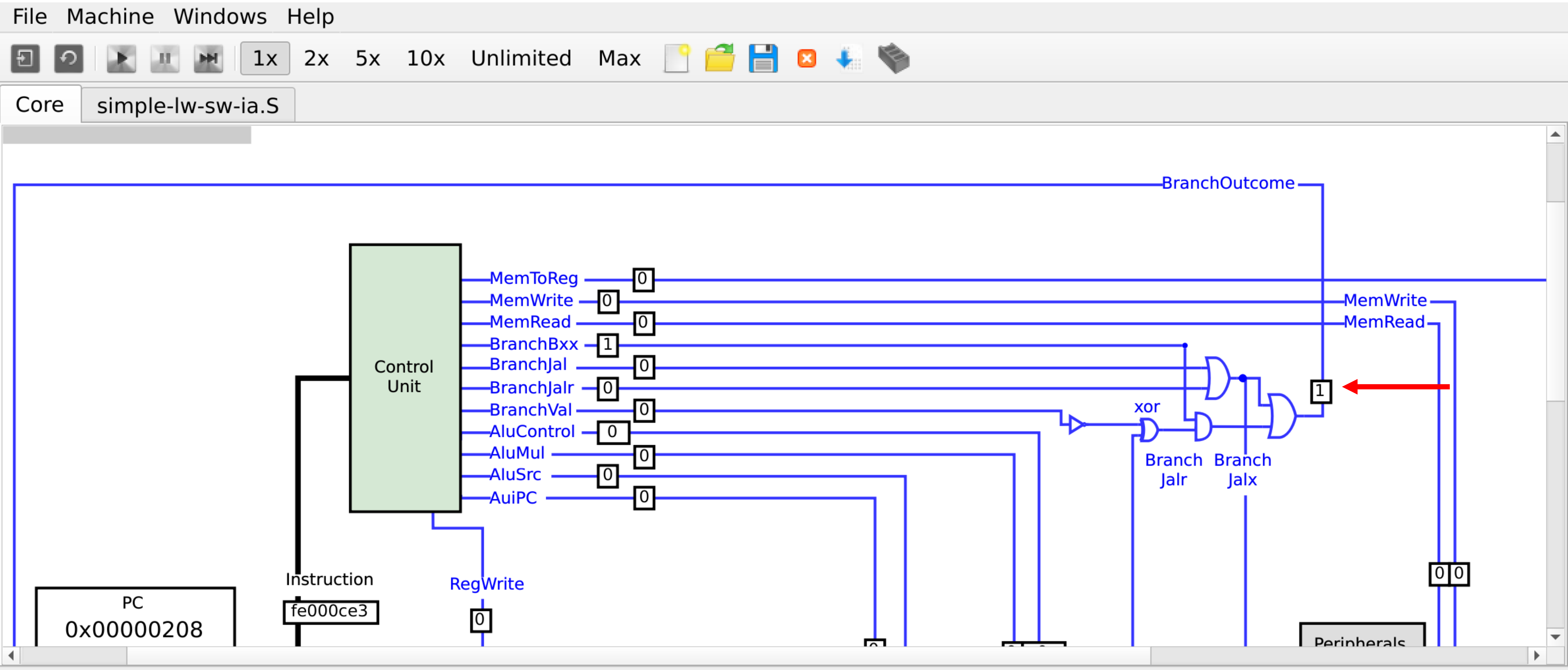
# Branch Control



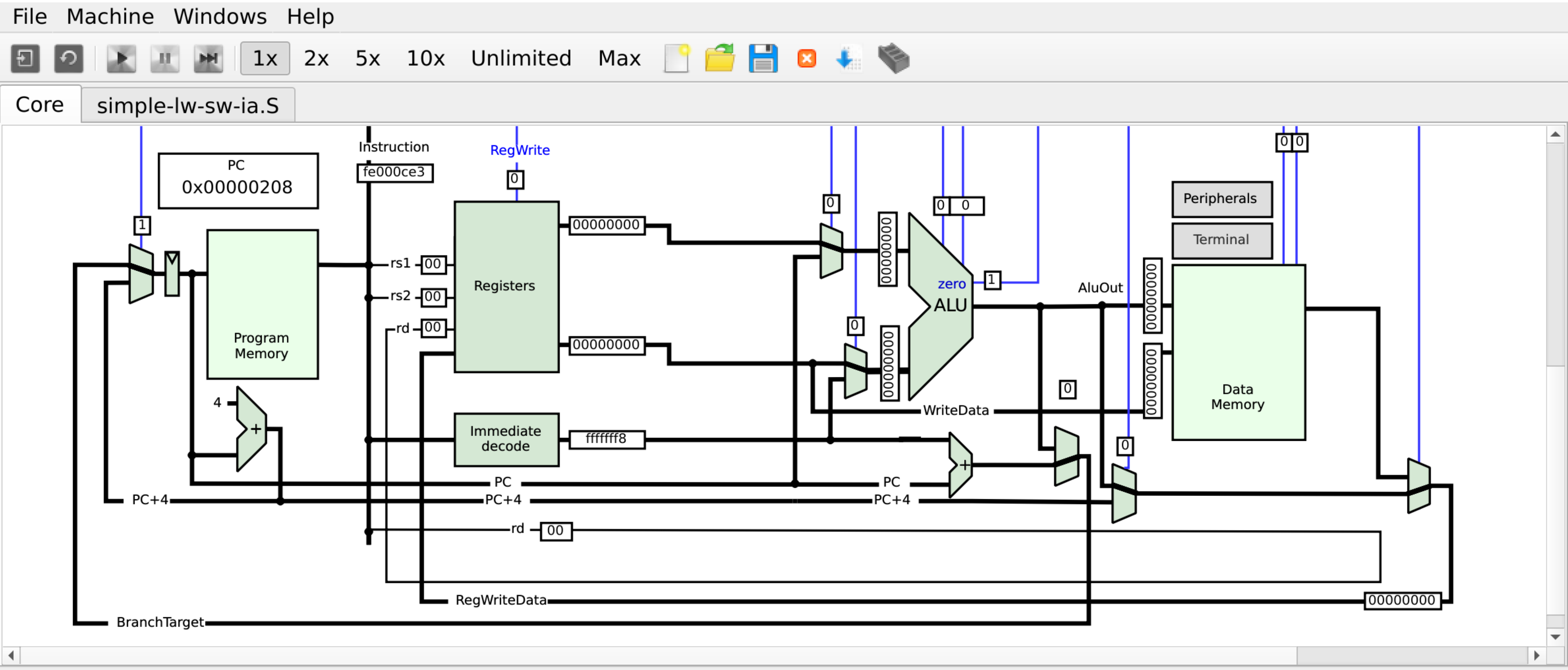
# Branch Control



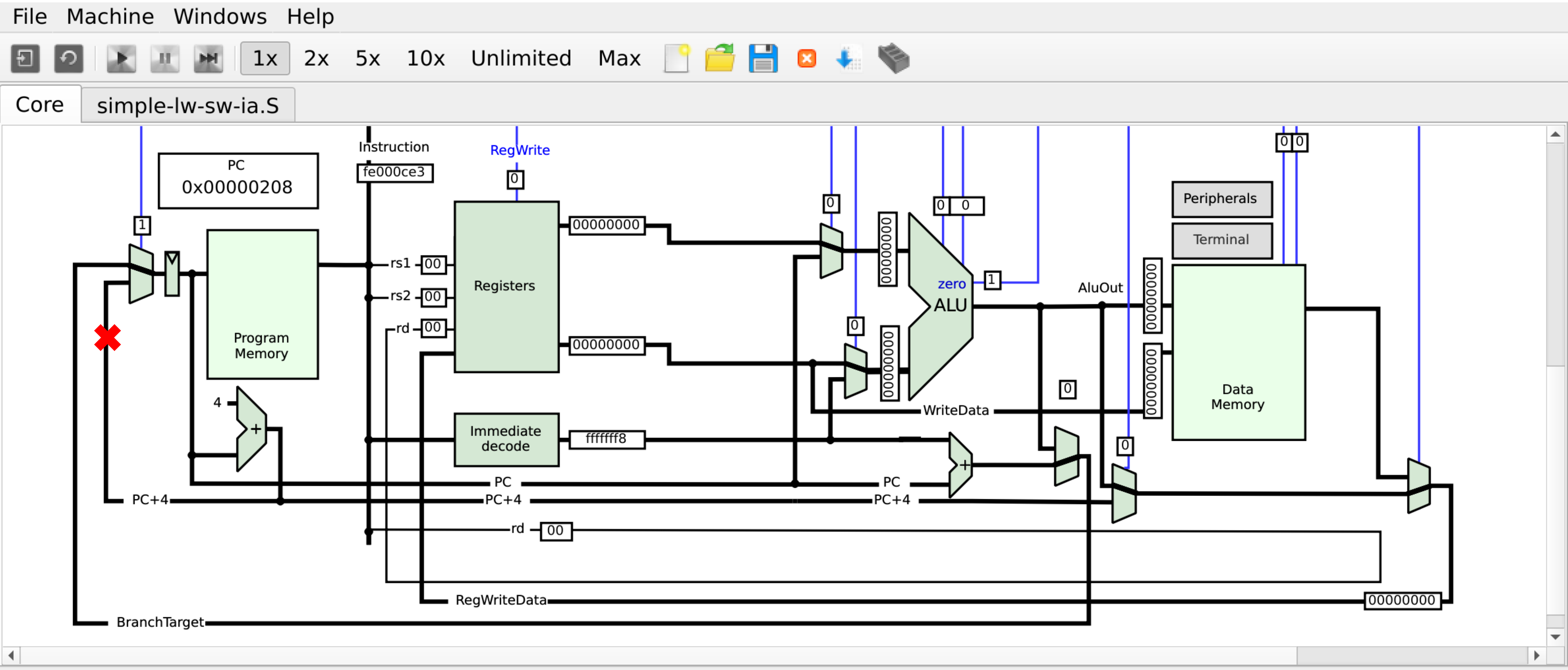
# Branch Control



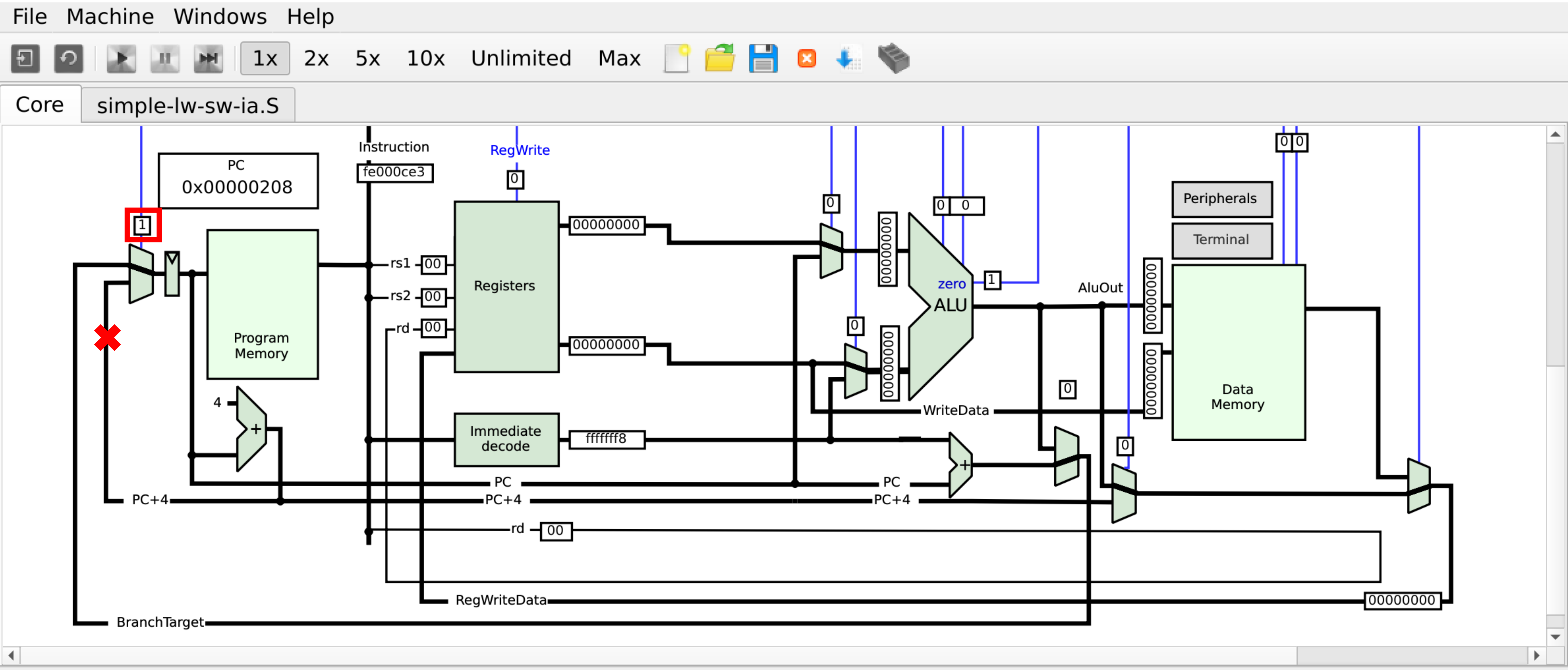
# Program Counter Update



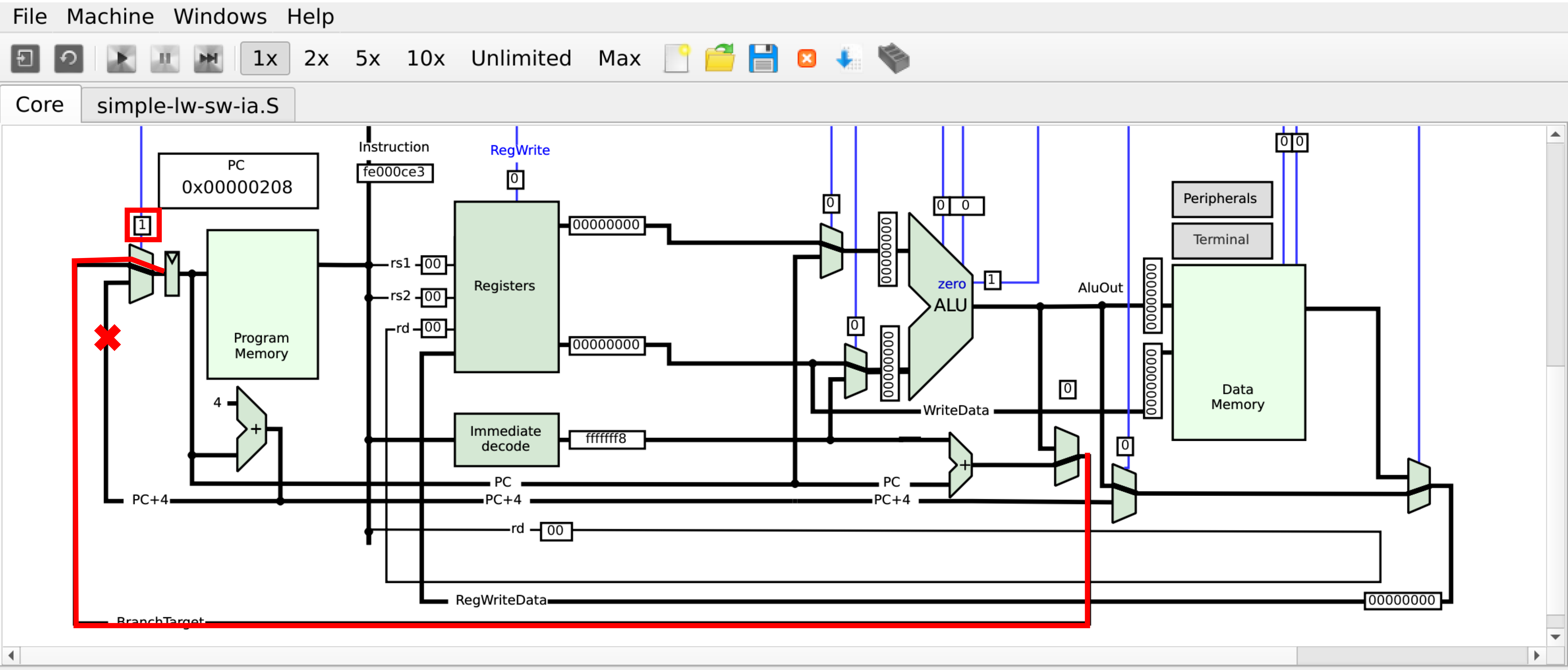
# Program Counter Update



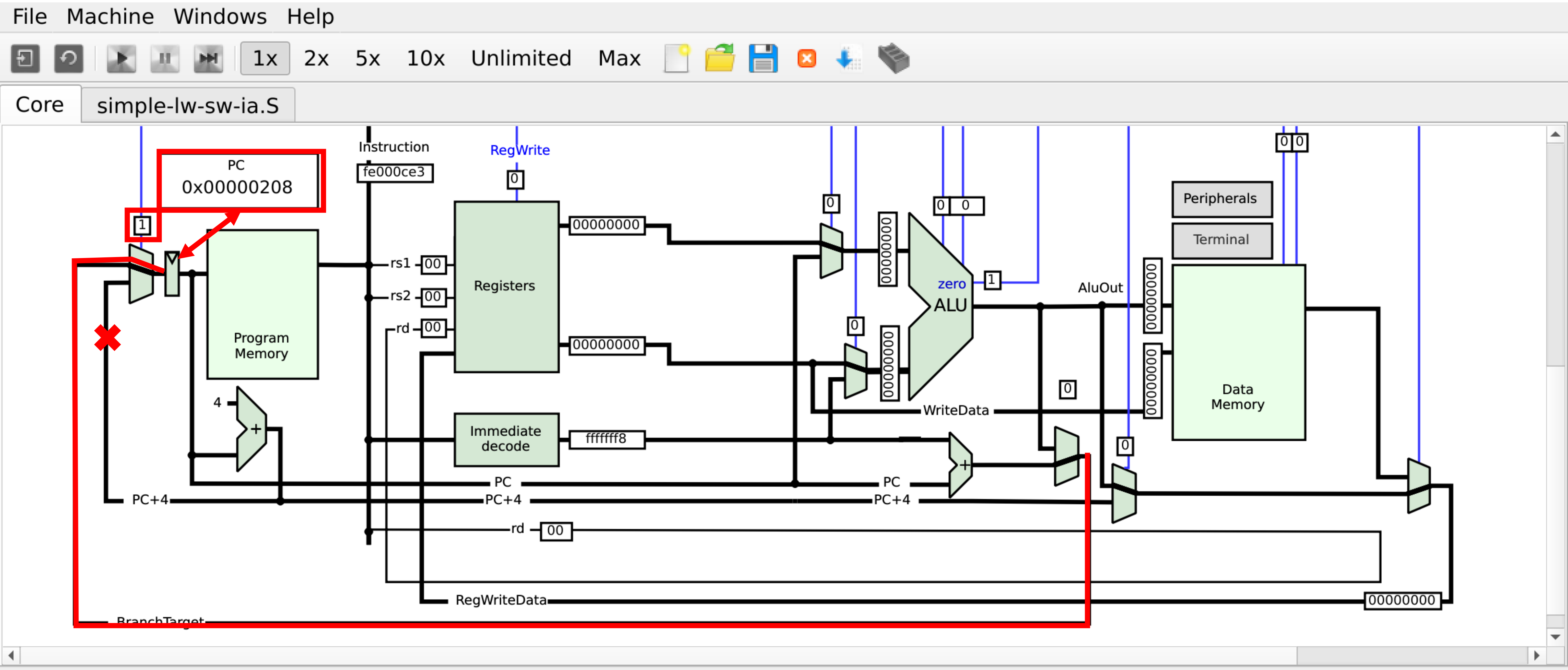
# Program Counter Update



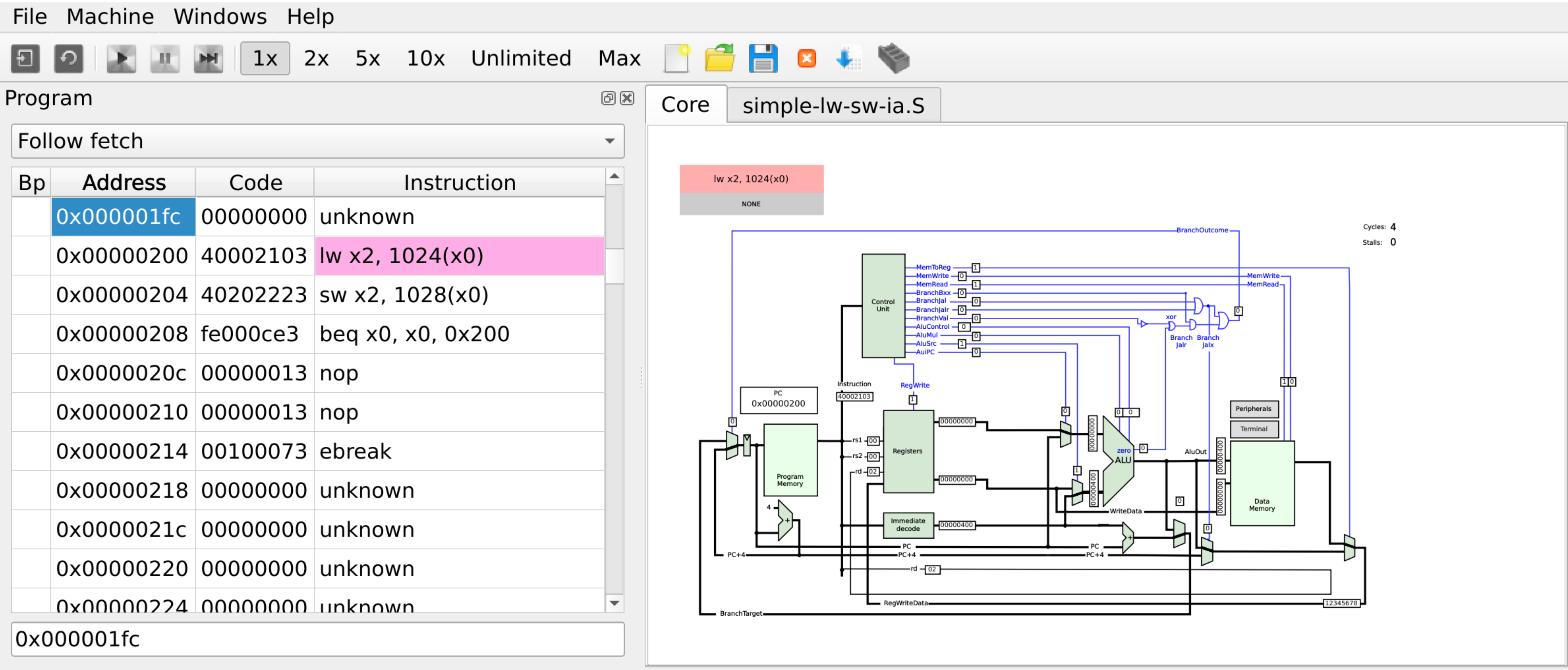
# Program Counter Update



# Program Counter Update



# Branch Result

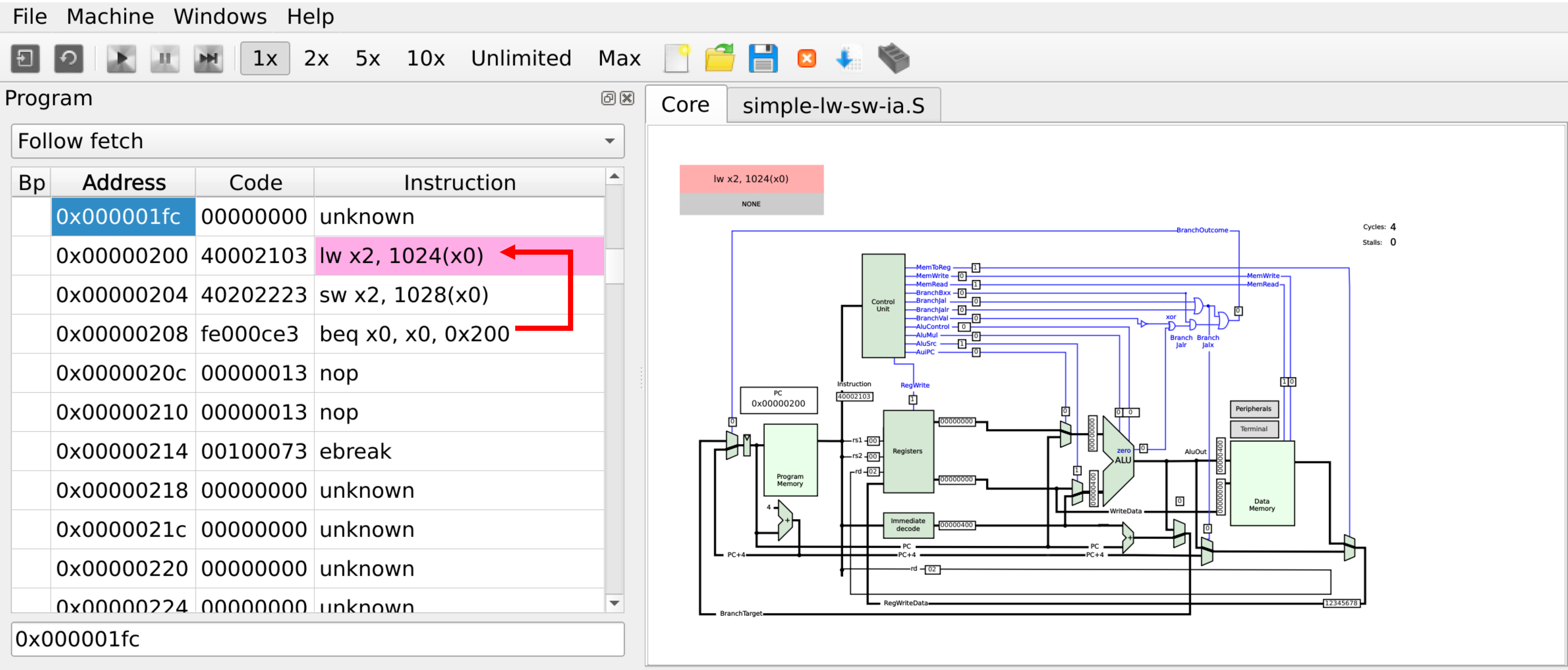


Follow fetch

| Bp | Address | Code | Instruction |
|---|---|---|---|
| | 0x000001fc | 00000000 | unknown |
| | 0x00000200 | 40002103 | lw x2, 1024(x0) |
| | 0x00000204 | 40202223 | sw x2, 1028(x0) |
| | 0x00000208 | fe000ce3 | beq x0, x0, 0x200 |
| | 0x0000020c | 00000013 | nop |
| | 0x00000210 | 00000013 | nop |
| | 0x00000214 | 00100073 | ebreak |
| | 0x00000218 | 00000000 | unknown |
| | 0x0000021c | 00000000 | unknown |
| | 0x00000220 | 00000000 | unknown |
| | 0x00000224 | 00000000 | unknown |

0x000001fc

# Branch Result



Program

Follow fetch

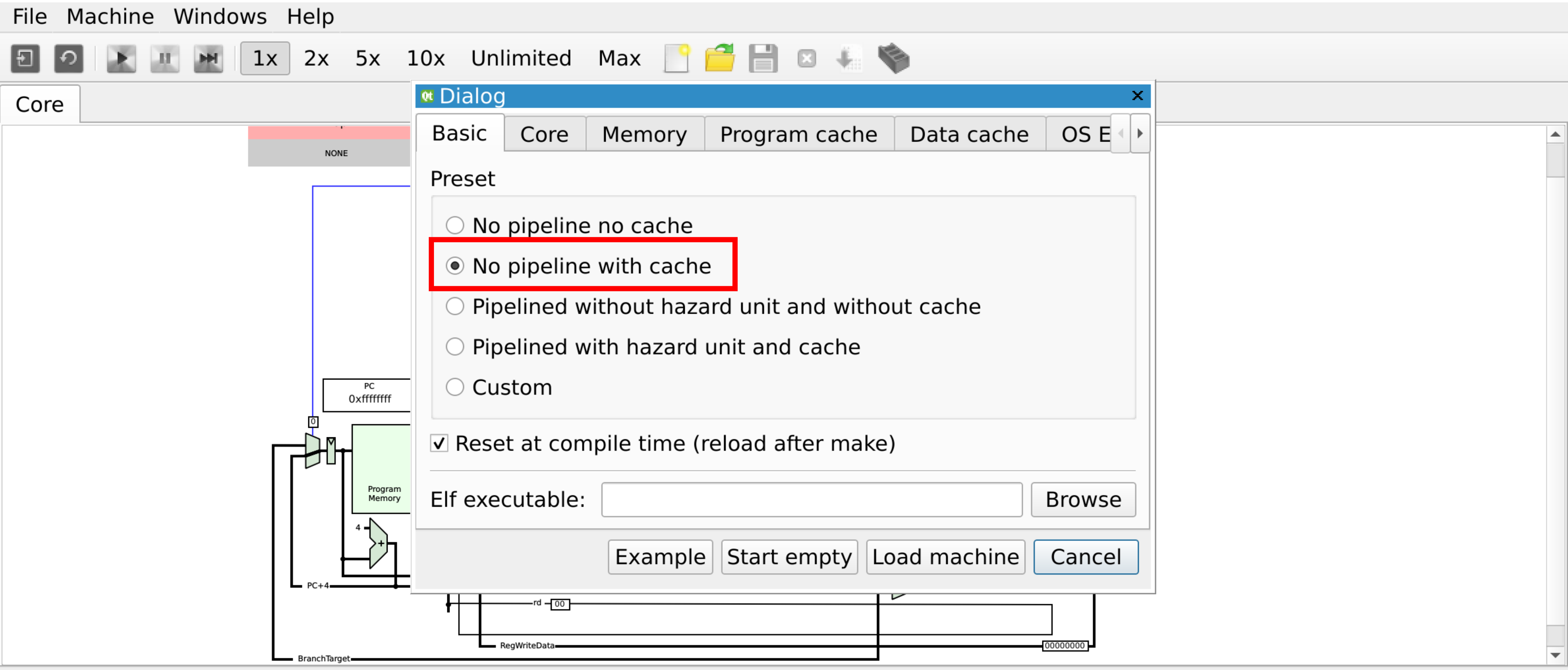
| Bp | Address | Code | Instruction |
|---|---|---|---|
| | 0x000001fc | 00000000 | unknown |
| | 0x00000200 | 40002103 | lw x2, 1024(x0) |
| | 0x00000204 | 40202223 | sw x2, 1028(x0) |
| | 0x00000208 | fe000ce3 | beq x0, x0, 0x200 |
| | 0x0000020c | 00000013 | nop |
| | 0x00000210 | 00000013 | nop |
| | 0x00000214 | 00100073 | ebreak |
| | 0x00000218 | 00000000 | unknown |
| | 0x0000021c | 00000000 | unknown |
| | 0x00000220 | 00000000 | unknown |
| | 0x00000224 | 00000000 | unknown |

0x000001fc

# Memory Cache

Cache Performance
Shape Configuration

# Config



Dialog

Basic | Core | Memory | Program cache | Data cache | OS E

Preset

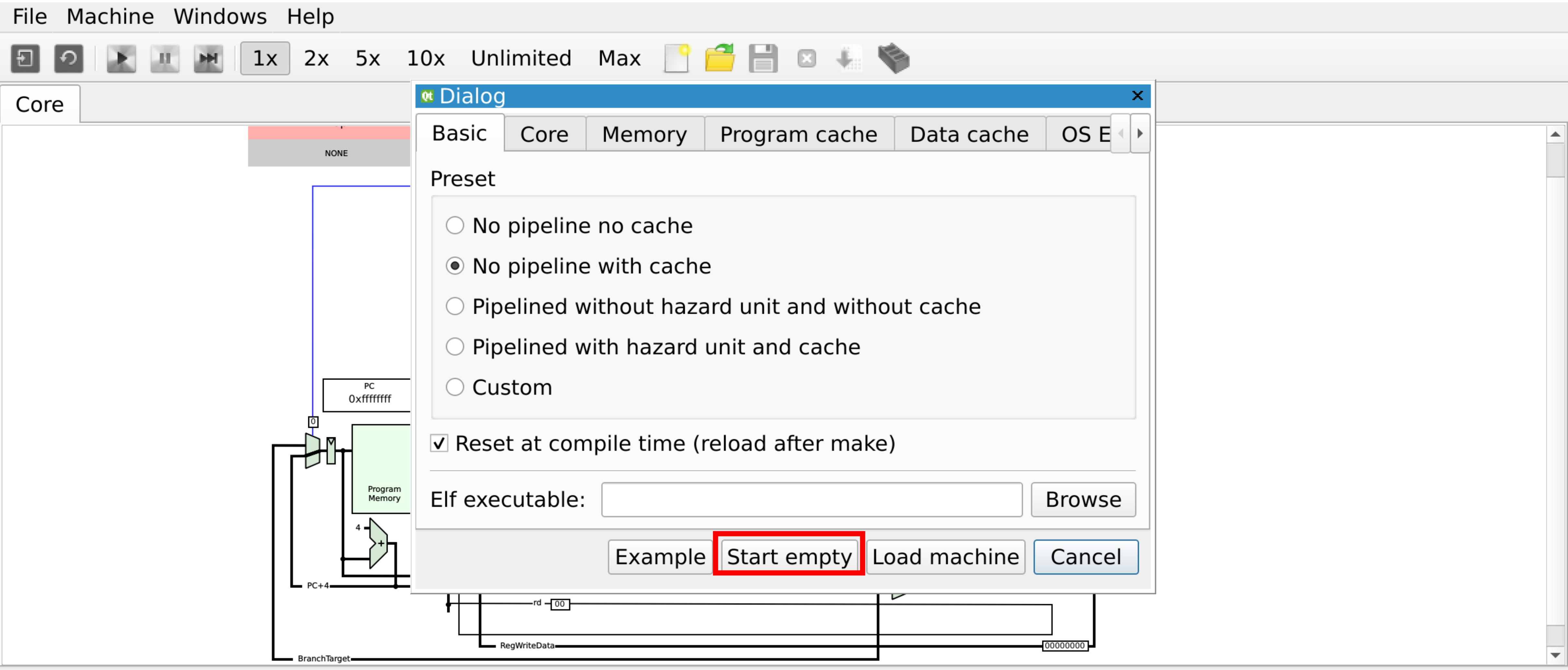
- ○ No pipeline no cache
- ● No pipeline with cache
- ○ Pipelined without hazard unit and without cache
- ○ Pipelined with hazard unit and cache
- ○ Custom

✓ Reset at compile time (reload after make)

Elf executable: Browse

Example | Start empty | Load machine | Cancel

# Config



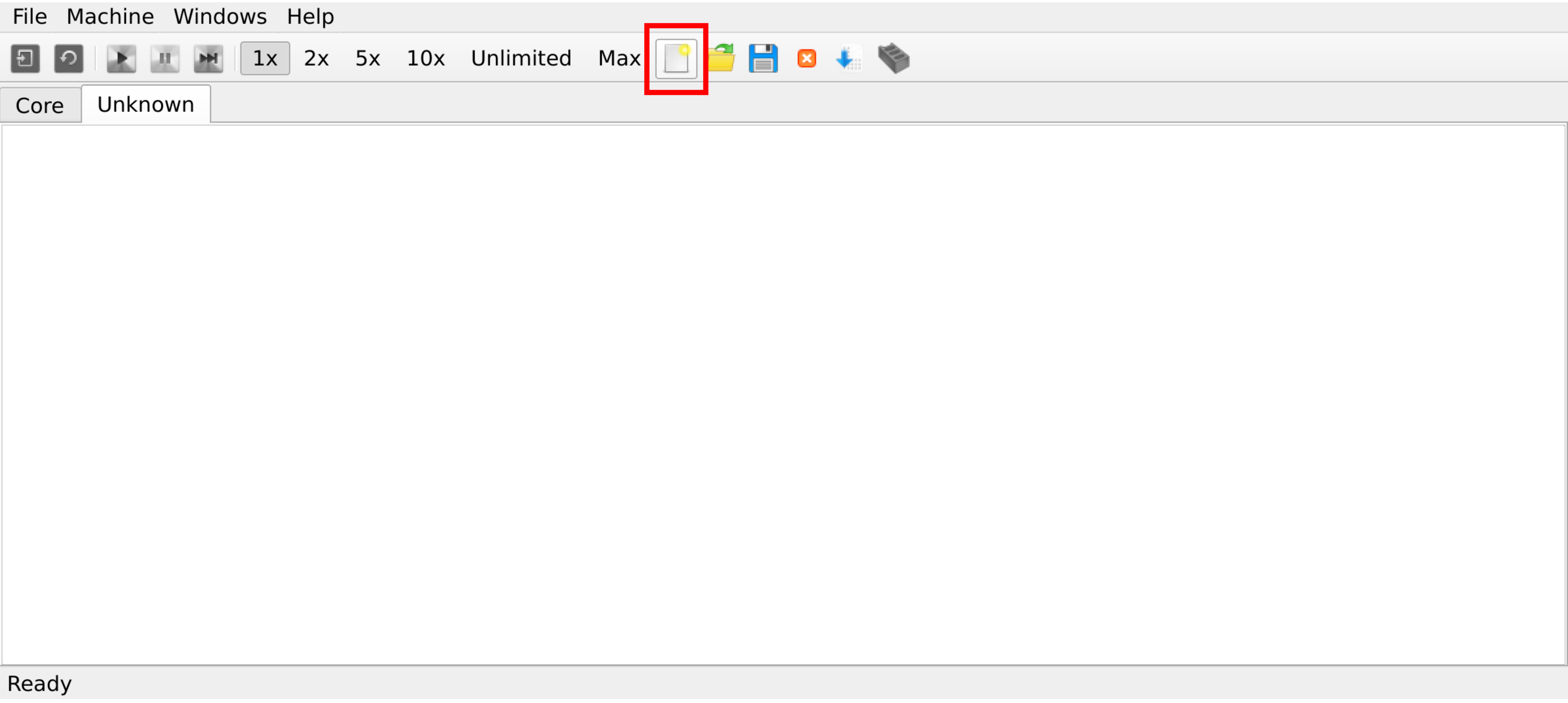
Preset

- No pipeline no cache
- [✓ No pipeline with cache]
- Pipelined without hazard unit and without cache
- Pipelined with hazard unit and cache
- Custom

✓ Reset at compile time (reload after make)

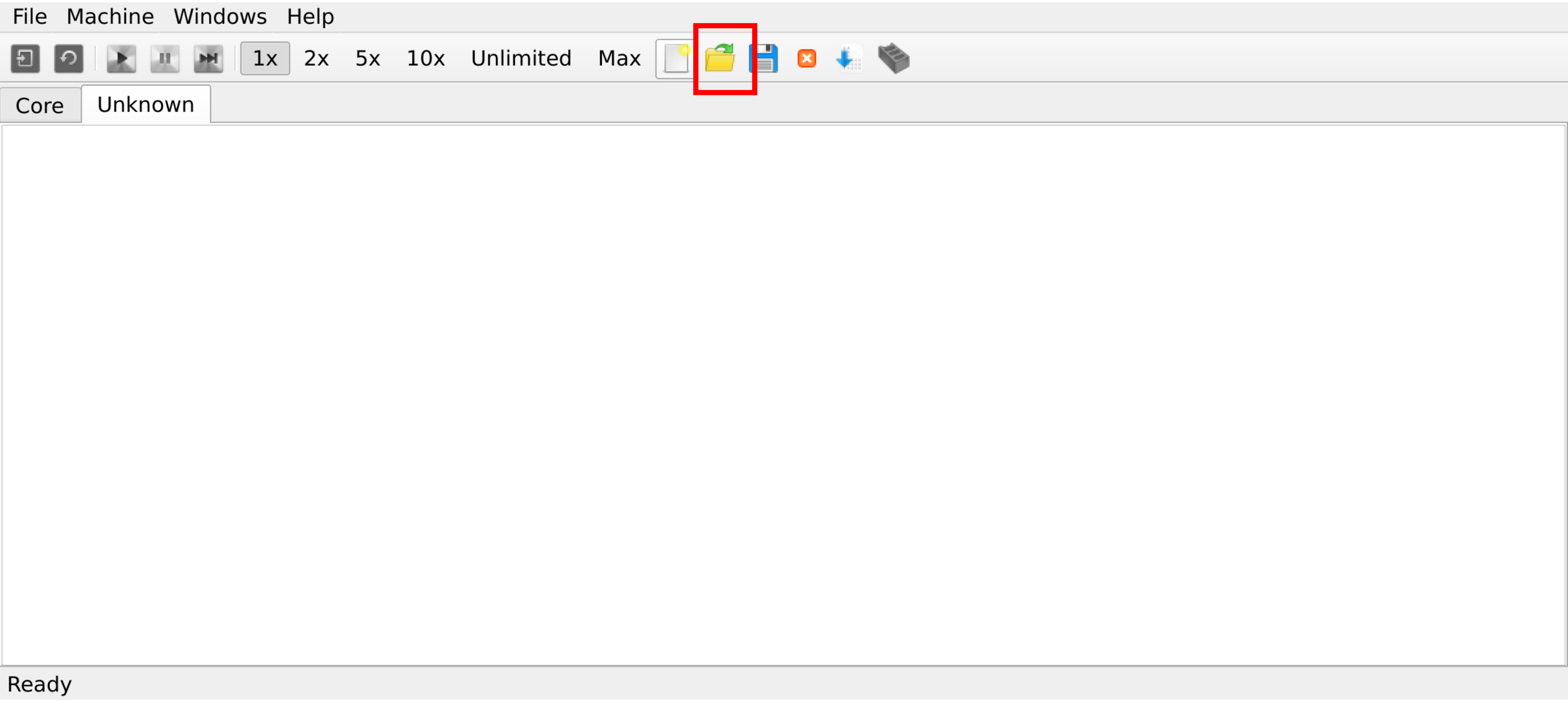
Elf executable:

# Editor & Assembler

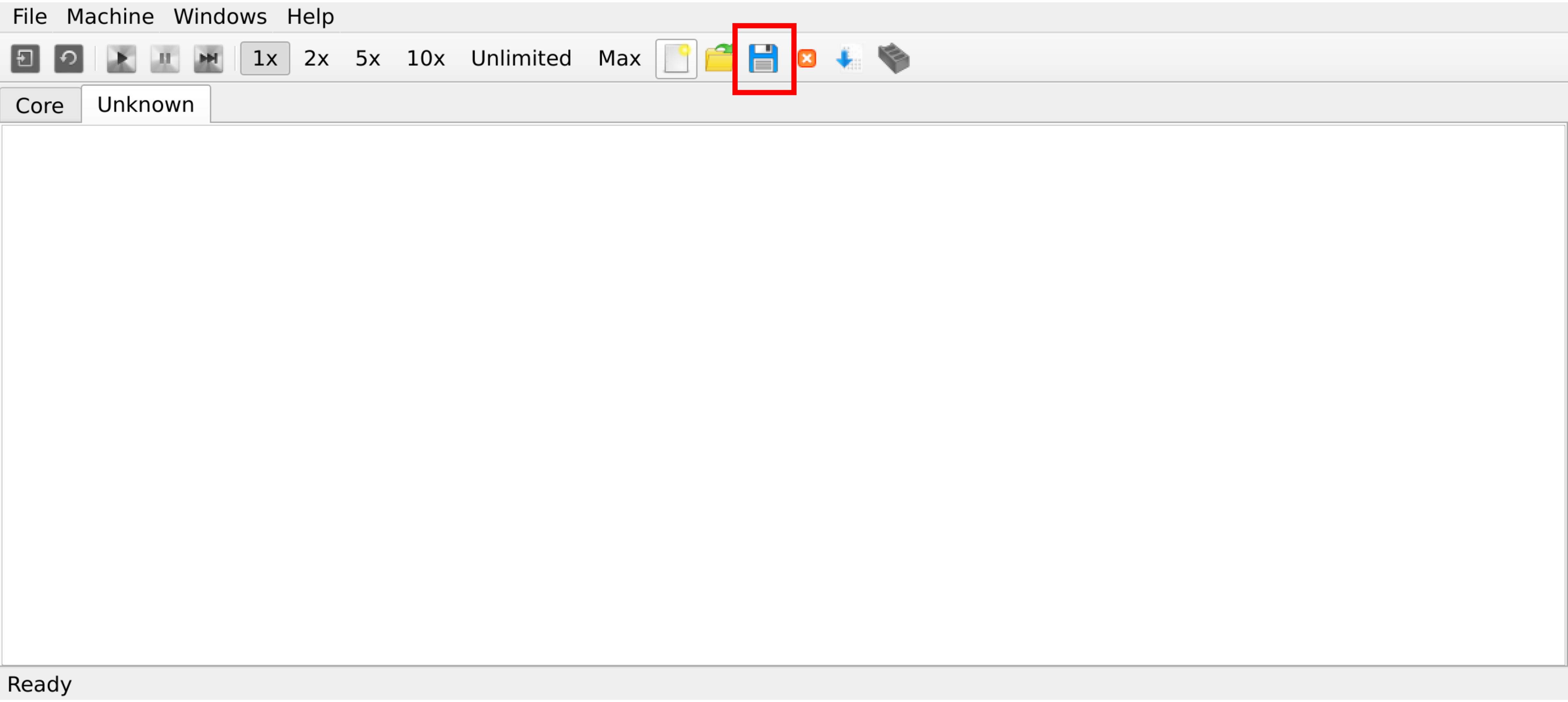


Ready

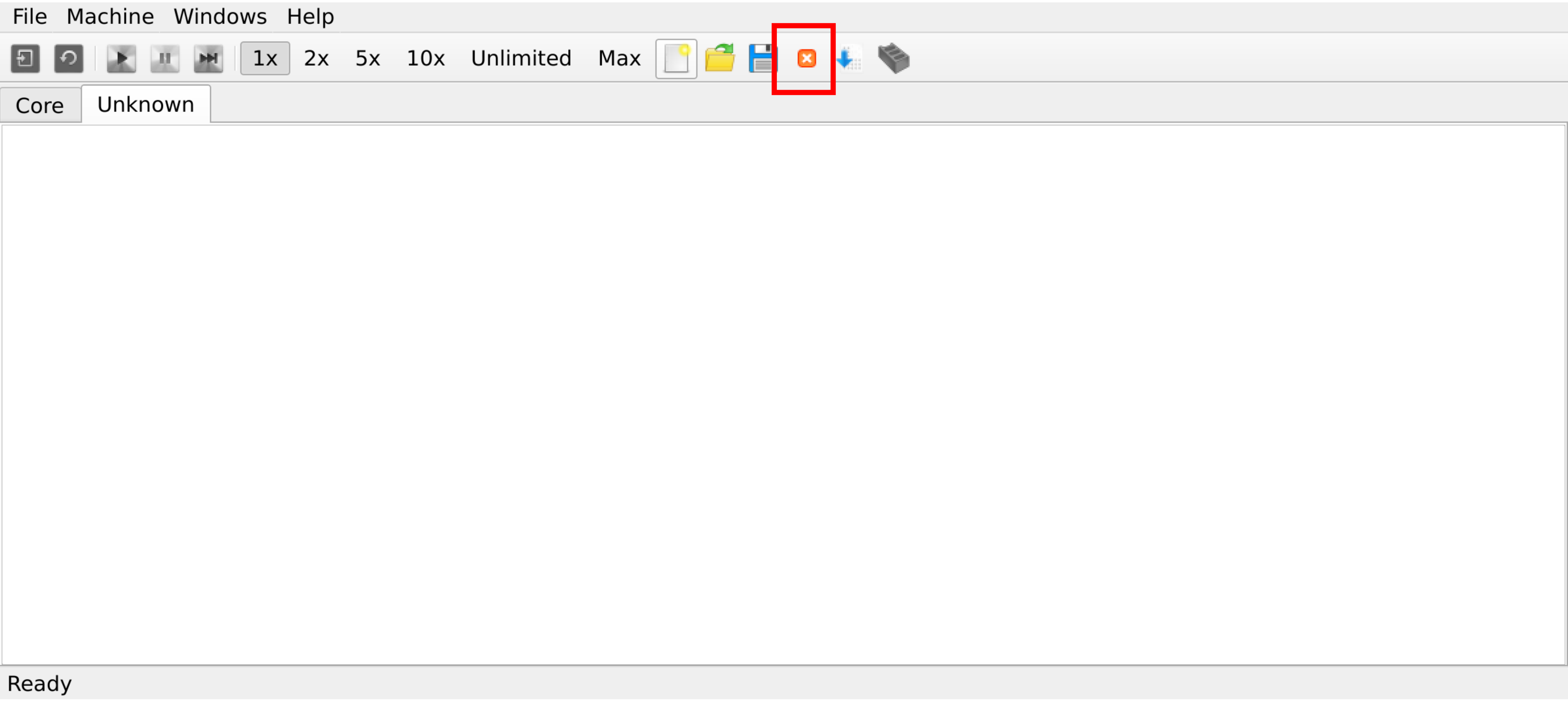
# Editor & Assembler



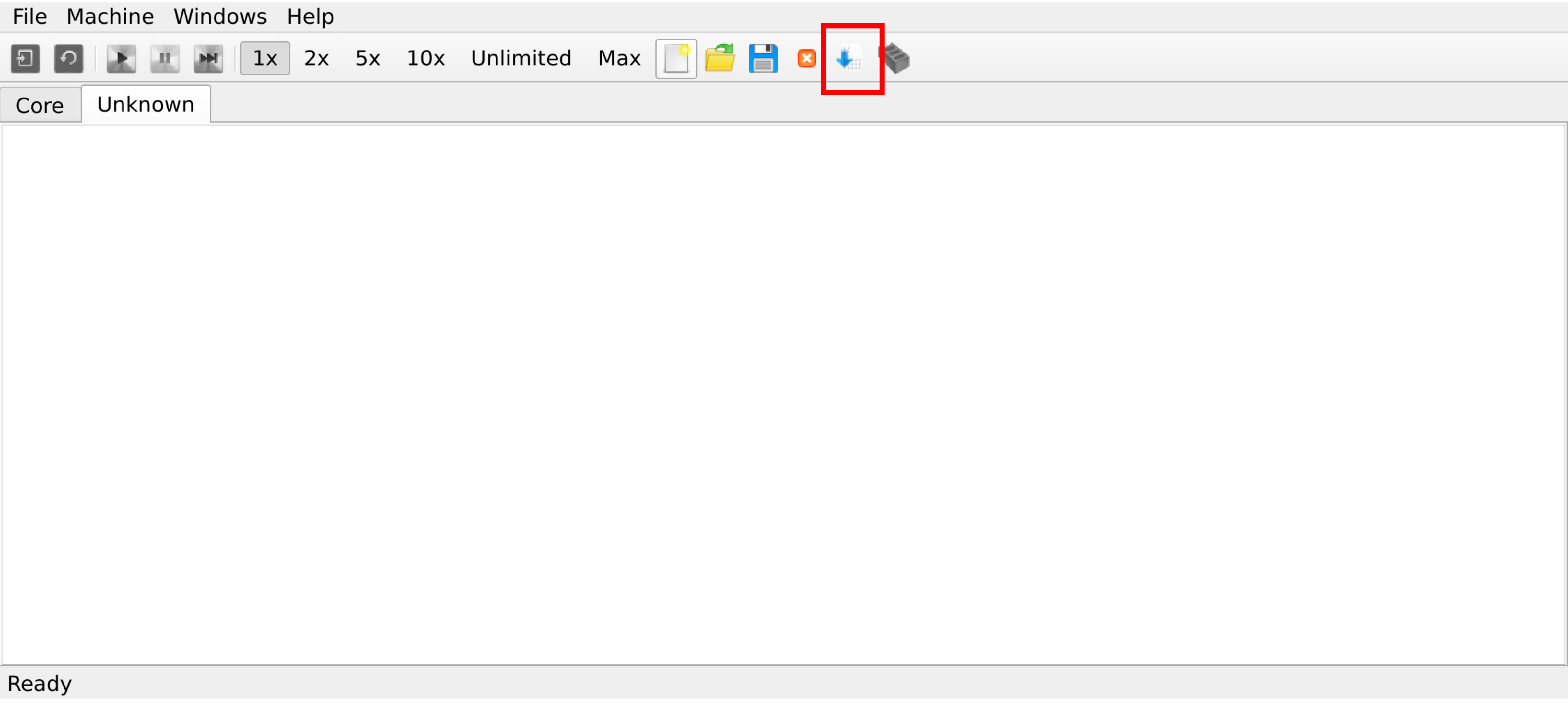
# Editor & Assembler



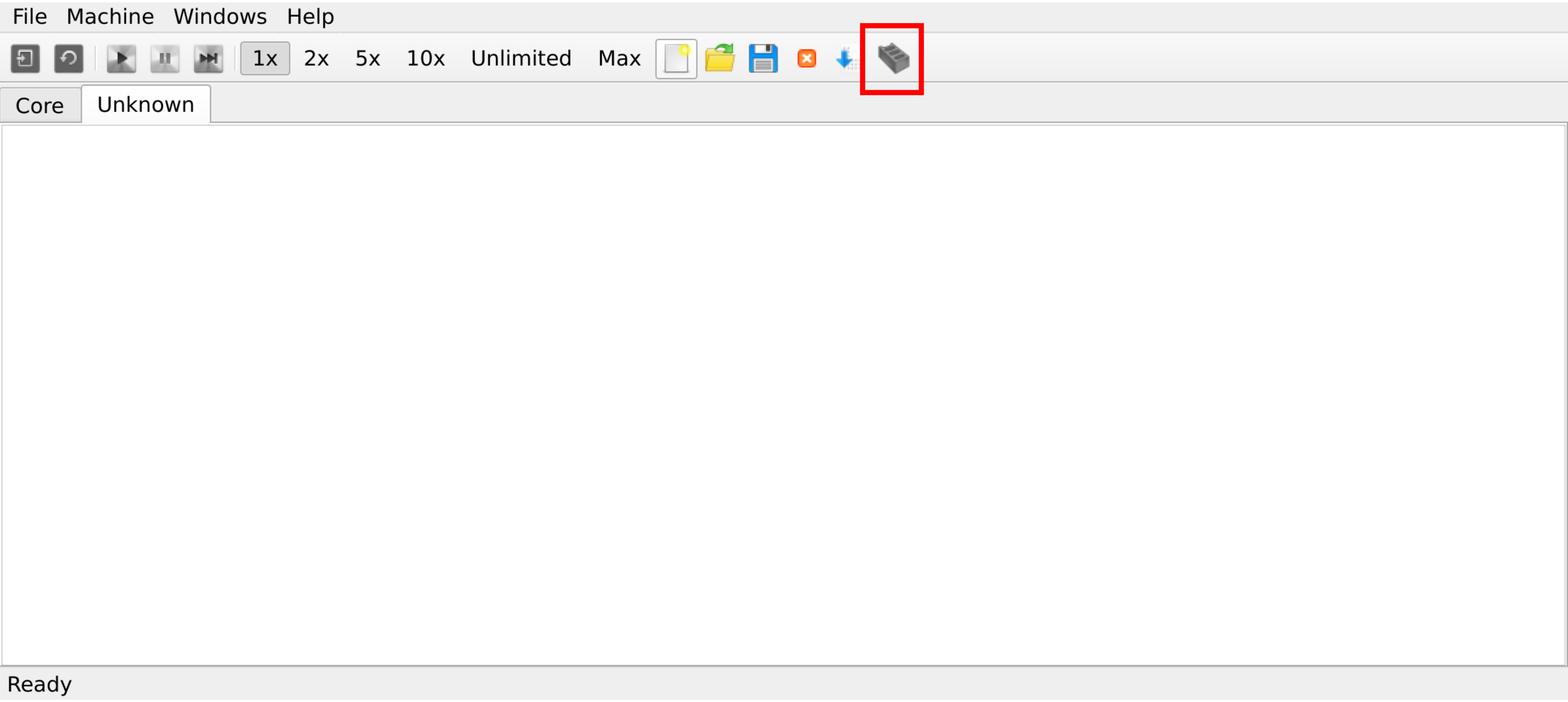
# Editor & Assembler



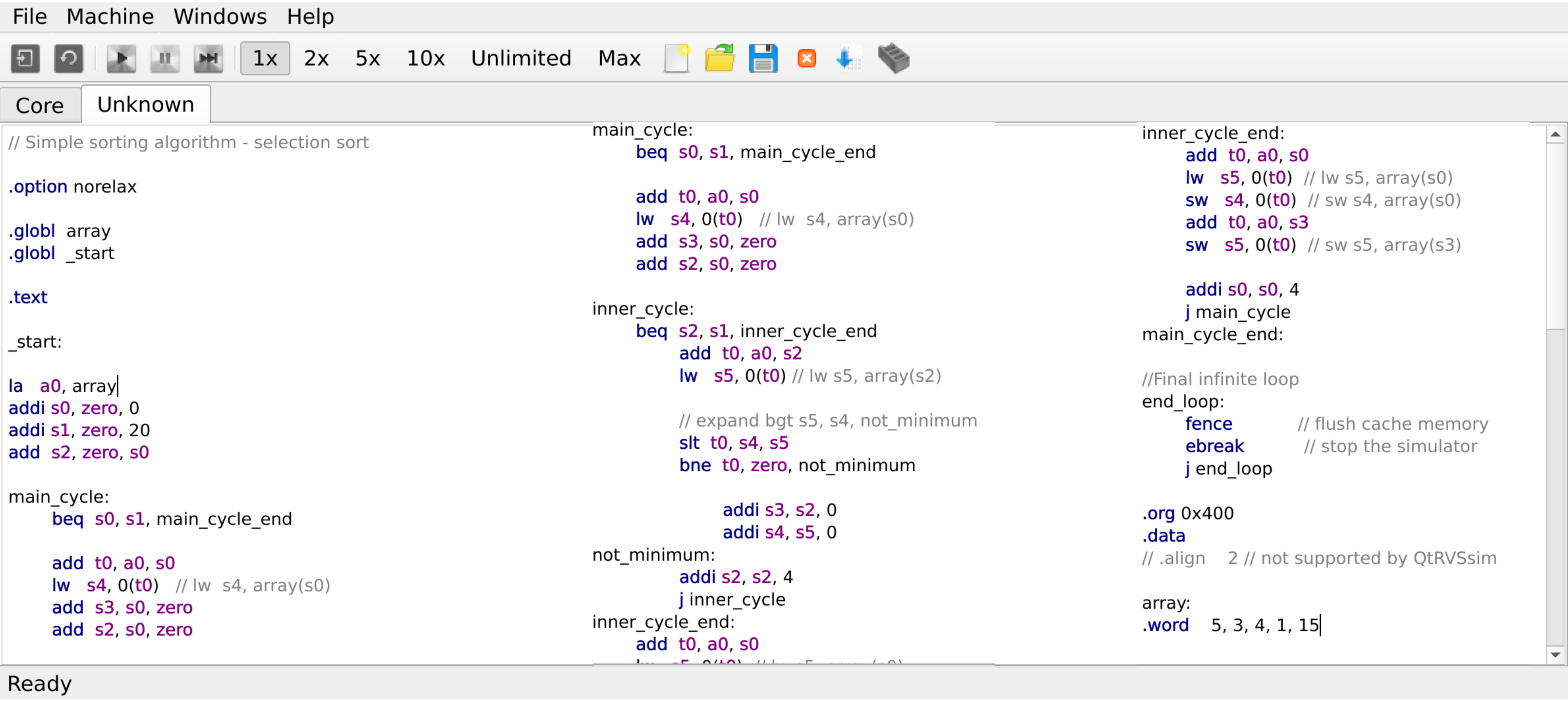
# Editor & Assembler



# Editor & Assembler



# Example: Selection Sort



```
// Simple sorting algorithm - selection sort

.option norelax

.globl  array
.globl  _start

.text

_start:

la   a0, array
addi s0, zero, 0
addi s1, zero, 20
add  s2, zero, s0

main_cycle:
     beq  s0, s1, main_cycle_end

     add  t0, a0, s0
     lw   s4, 0(t0)   // lw  s4, array(s0)
     add  s3, s0, zero
     add  s2, s0, zero
```

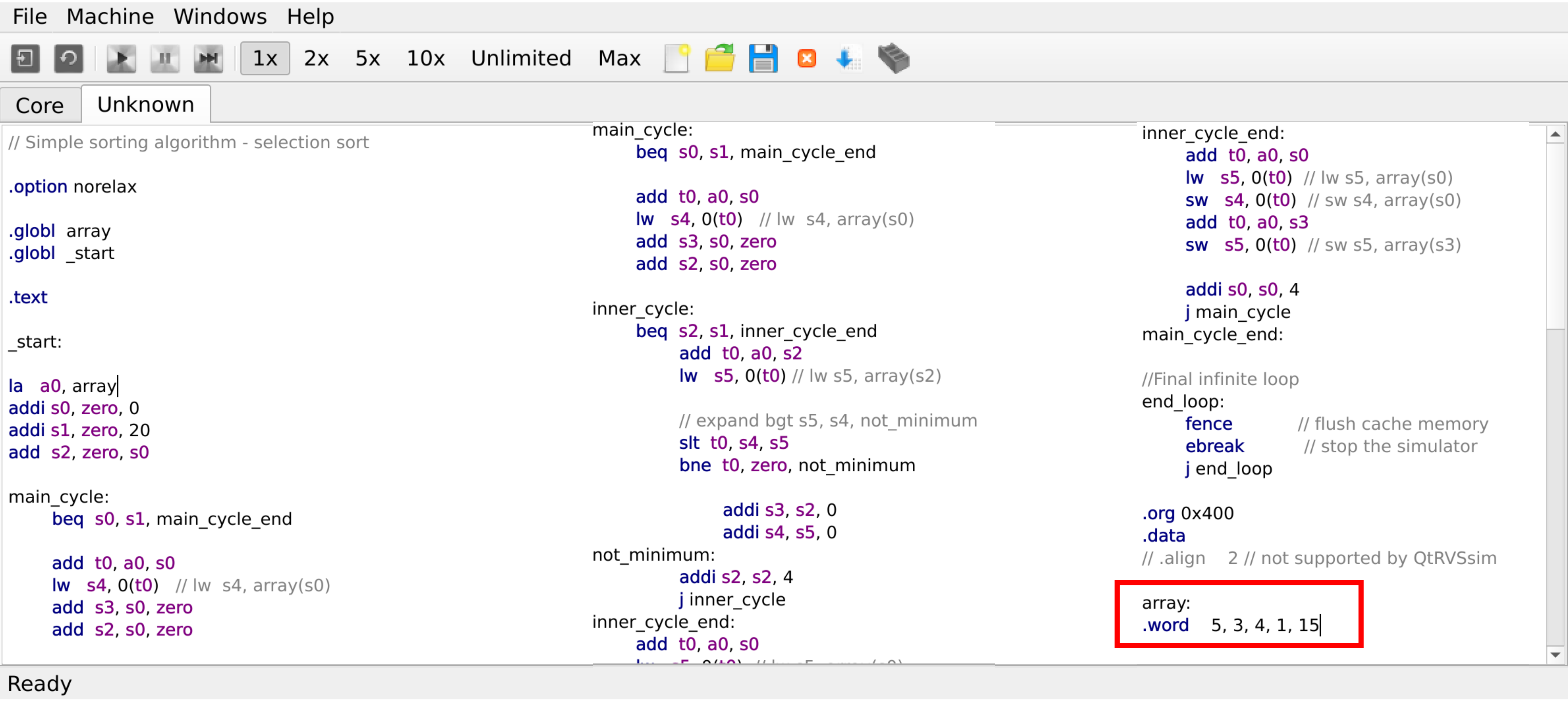
```
main_cycle:
     beq  s0, s1, main_cycle_end

     add  t0, a0, s0
     lw   s4, 0(t0)   // lw  s4, array(s0)
     add  s3, s0, zero
     add  s2, s0, zero
```

```
inner_cycle:
     beq  s2, s1, inner_cycle_end
          add  t0, a0, s2
          lw   s5, 0(t0) // lw s5, array(s2)

          // expand bgt s5, s4, not_minimum
          slt  t0, s4, s5
          bne  t0, zero, not_minimum

               addi s3, s2, 0
               addi s4, s5, 0
not_minimum:
          addi s2, s2, 4
          j inner_cycle
inner_cycle_end:
     add  t0, a0, s0
```

```
inner_cycle_end:
```

```
     add  t0, a0, s0
     lw   s5, 0(t0)  // lw s5, array(s0)
     sw   s4, 0(t0)  // sw s4, array(s0)
     add  t0, a0, s3
     sw   s5, 0(t0)  // sw s5, array(s3)

     addi s0, s0, 4
     j main_cycle
main_cycle_end:

//Final infinite loop
end_loop:
     fence          // flush cache memory
     ebreak         // stop the simulator
     j end_loop

.org 0x400
.data
// .align    2 // not supported by QtRVSsim

array:
.word    5, 3, 4, 1, 15
```

# Example: Selection Sort

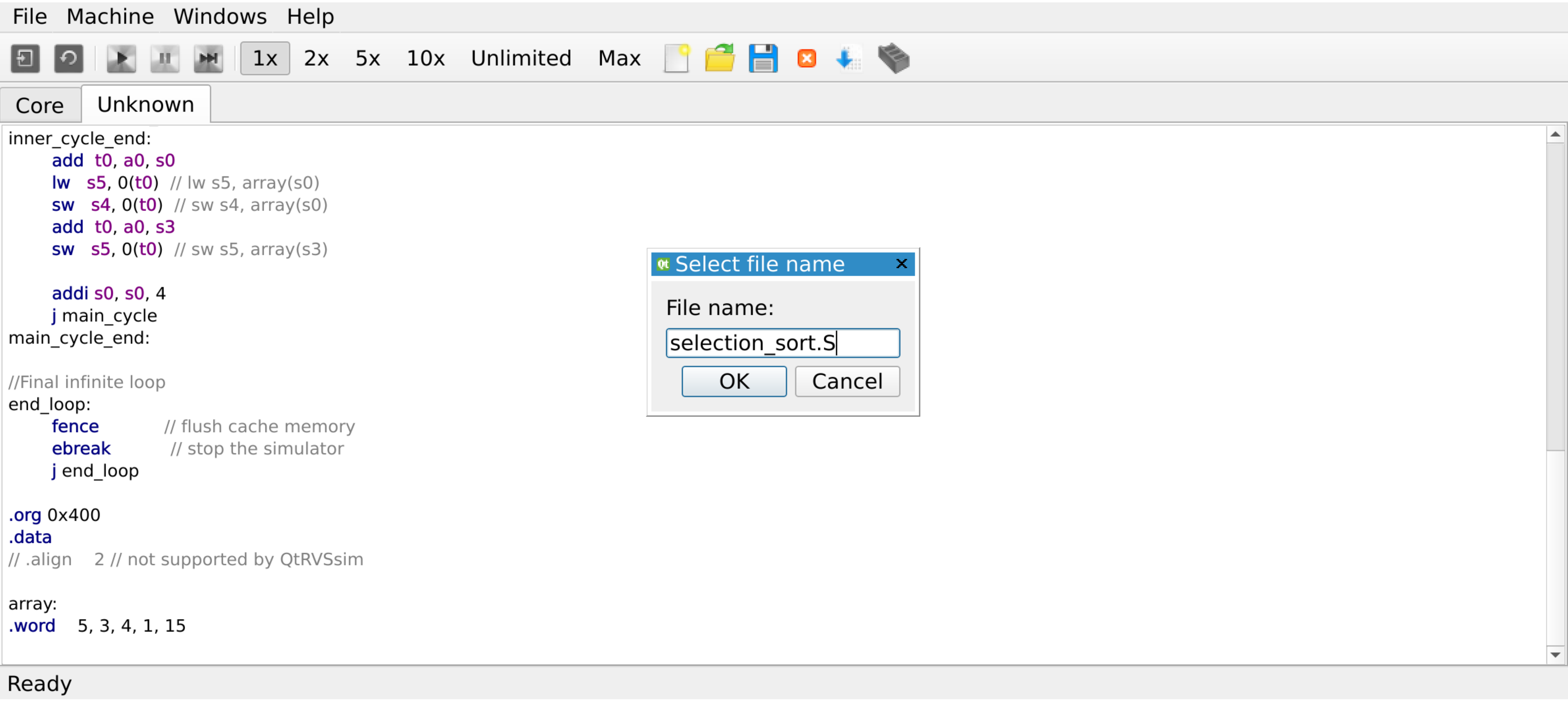


```
// Simple sorting algorithm - selection sort

.option norelax

.globl  array
.globl  _start

.text

_start:

la   a0, array
addi s0, zero, 0
addi s1, zero, 20
add  s2, zero, s0

main_cycle:
      beq  s0, s1, main_cycle_end

      add  t0, a0, s0
      lw   s4, 0(t0)   // lw  s4, array(s0)
      add  s3, s0, zero
      add  s2, s0, zero
```

```
main_cycle:
      beq  s0, s1, main_cycle_end

      add  t0, a0, s0
      lw   s4, 0(t0)   // lw  s4, array(s0)
      add  s3, s0, zero
      add  s2, s0, zero

inner_cycle:
      beq  s2, s1, inner_cycle_end
            add  t0, a0, s2
            lw   s5, 0(t0) // lw s5, array(s2)

            // expand bgt s5, s4, not_minimum
            slt  t0, s4, s5
            bne  t0, zero, not_minimum

                  addi s3, s2, 0
                  addi s4, s5, 0
not_minimum:
            addi s2, s2, 4
            j inner_cycle
inner_cycle_end:
      add  t0, a0, s0
```

```
inner_cycle_end:
      add  t0, a0, s0
      lw   s5, 0(t0)  // lw s5, array(s0)
      sw   s4, 0(t0)  // sw s4, array(s0)
      add  t0, a0, s3
      sw   s5, 0(t0)  // sw s5, array(s3)

      addi s0, s0, 4
      j main_cycle
main_cycle_end:

//Final infinite loop
end_loop:
      fence          // flush cache memory
      ebreak         // stop the simulator
      j end_loop

.org 0x400
.data
// .align    2 // not supported by QtRVSsim

array:
.word    5, 3, 4, 1, 15
```

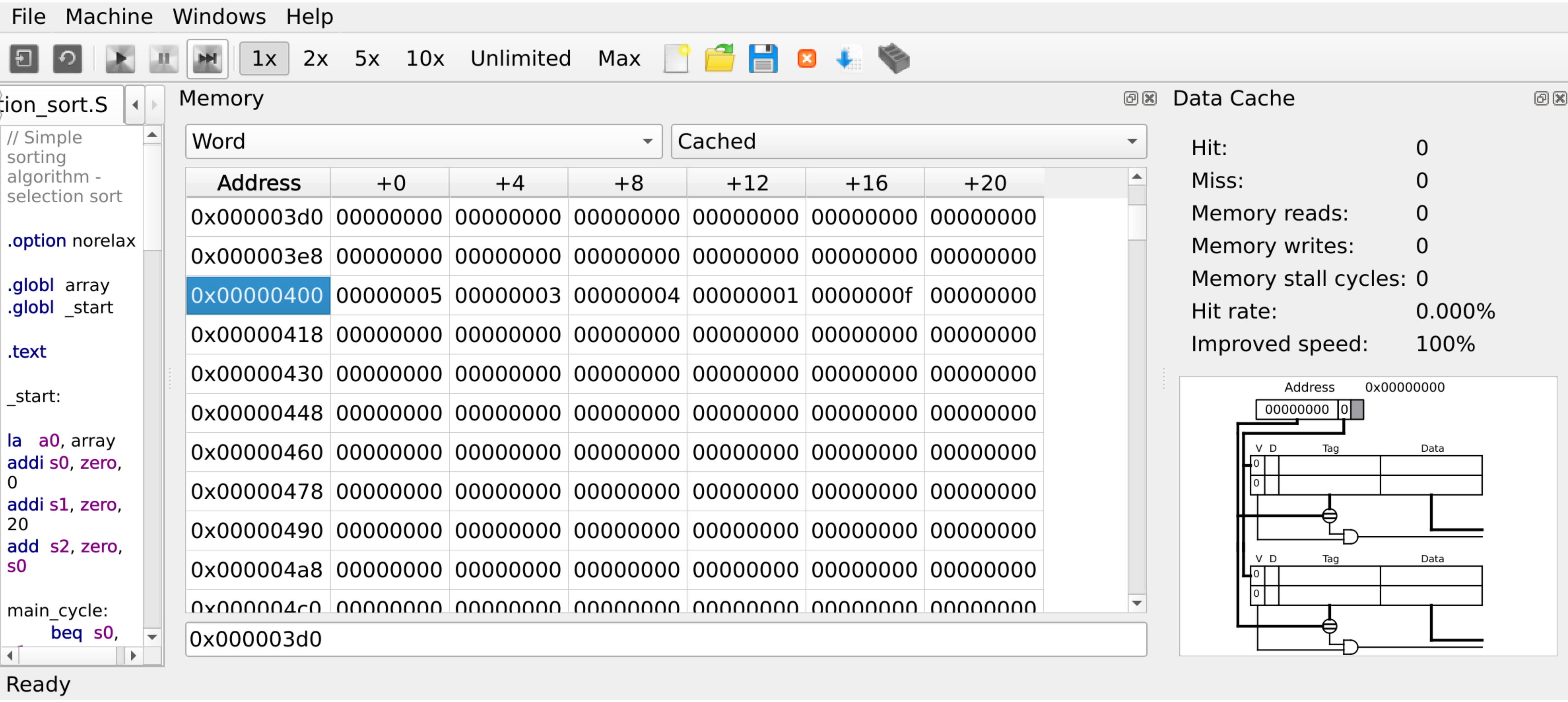
# Editor: Save File



```
inner_cycle_end:
      add  t0, a0, s0
      lw   s5, 0(t0)  // lw s5, array(s0)
      sw   s4, 0(t0)  // sw s4, array(s0)
      add  t0, a0, s3
      sw   s5, 0(t0)  // sw s5, array(s3)

      addi s0, s0, 4
      j main_cycle
main_cycle_end:

//Final infinite loop
end_loop:
      fence         // flush cache memory
      ebreak         // stop the simulator
      j end_loop

.org 0x400
.data
// .align    2 // not supported by QtRVSsim

array:
.word    5, 3, 4, 1, 15
```

File name: selection_sort.S

# Cache View



File Machine Windows Help

1x 2x 5x 10x Unlimited Max

tion_sort.S

```
// Simple
sorting
algorithm -
selection sort

.option norelax

.globl  array
.globl  _start

.text

_start:

la   a0, array
addi s0, zero,
0
addi s1, zero,
20
add  s2, zero,
s0

main_cycle:
        beq  s0,
```

Memory

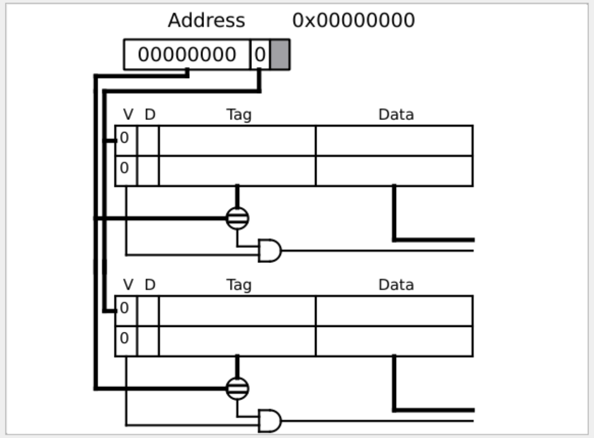
Word | Cached

| Address | +0 | +4 | +8 | +12 | +16 | +20 |
|---|---|---|---|---|---|---|
| 0x000003d0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000003e8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000400 | 00000005 | 00000003 | 00000004 | 00000001 | 0000000f | 00000000 |
| 0x00000418 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000430 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000448 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000460 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000478 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000490 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004a8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004c0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |

0x000003d0

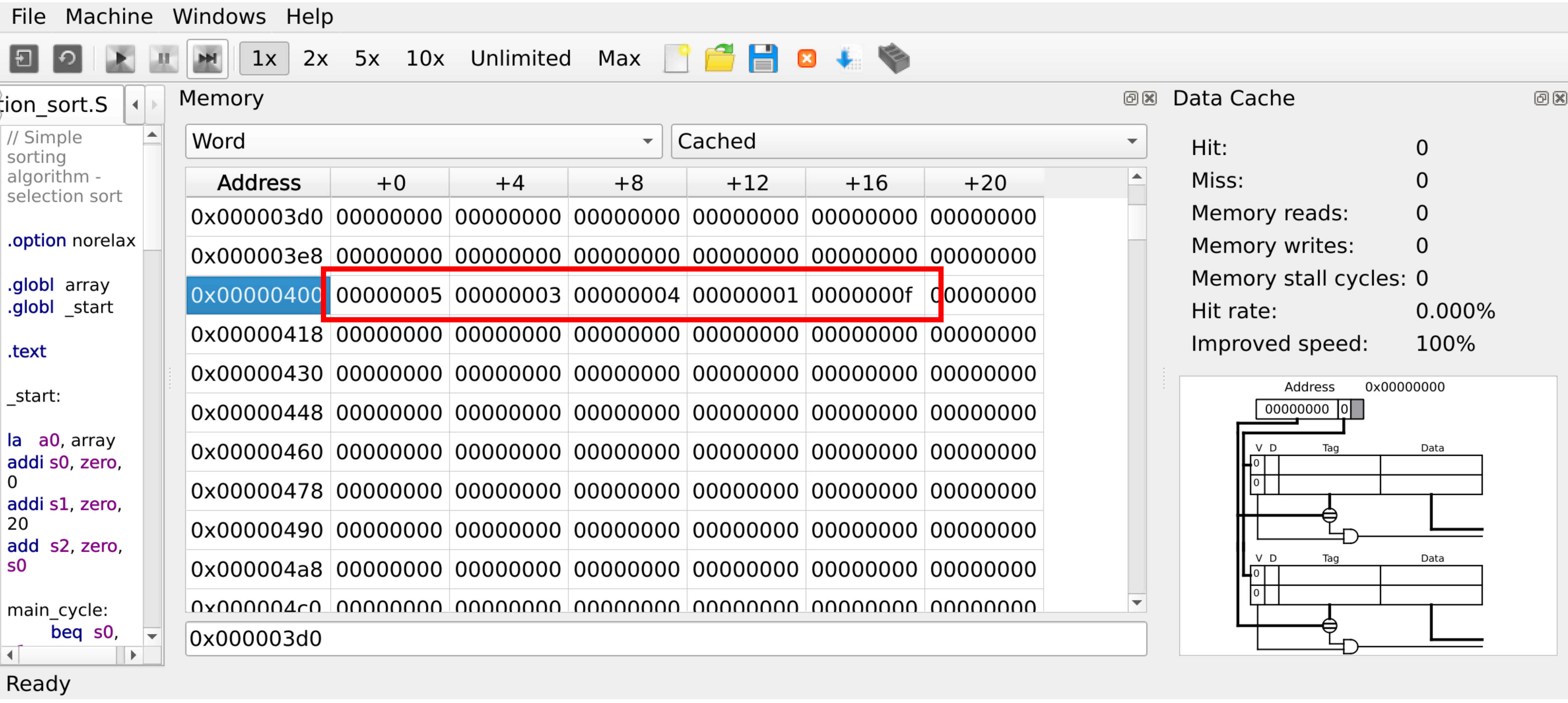
Data Cache

| | |
|---|---|
| Hit: | 0 |
| Miss: | 0 |
| Memory reads: | 0 |
| Memory writes: | 0 |
| Memory stall cycles: | 0 |
| Hit rate: | 0.000% |
| Improved speed: | 100% |



Ready

# Cache View



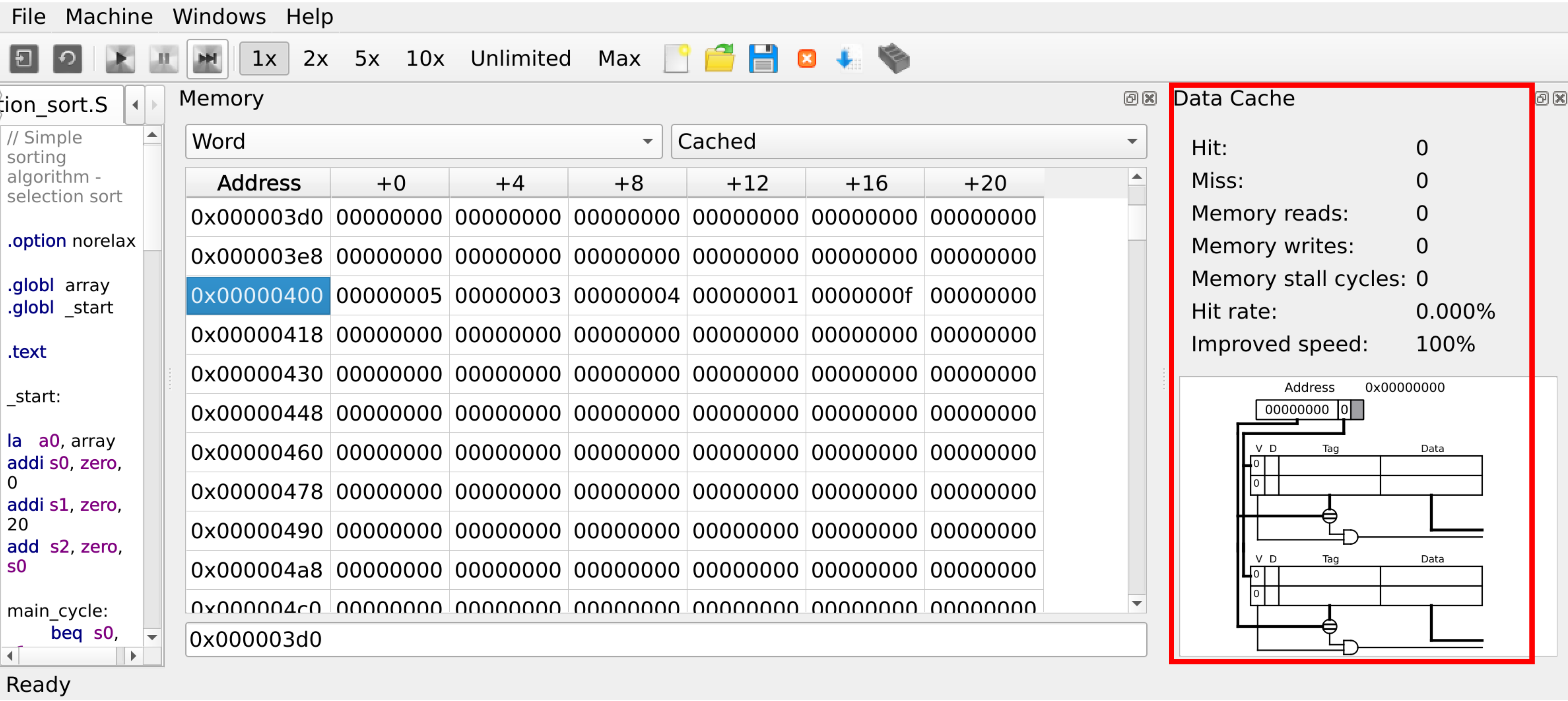
```
// Simple
sorting
algorithm -
selection sort

.option norelax

.globl  array
.globl  _start

.text

_start:

la   a0, array
addi s0, zero,
0
addi s1, zero,
20
add  s2, zero,
s0

main_cycle:
        beq  s0,
```

| Address | +0 | +4 | +8 | +12 | +16 | +20 |
|---|---|---|---|---|---|---|
| 0x000003d0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000003e8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000400 | 00000005 | 00000003 | 00000004 | 00000001 | 0000000f | 00000000 |
| 0x00000418 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000430 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000448 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000460 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000478 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000490 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004a8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004c0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |

0x000003d0

| | |
|---|---|
| Hit: | 0 |
| Miss: | 0 |
| Memory reads: | 0 |
| Memory writes: | 0 |
| Memory stall cycles: | 0 |
| Hit rate: | 0.000% |
| Improved speed: | 100% |

# Cache View



```
// Simple
sorting
algorithm -
selection sort

.option norelax

.globl  array
.globl  _start

.text

_start:

la   a0, array
addi s0, zero,
0
addi s1, zero,
20
add  s2, zero,
s0

main_cycle:
     beq  s0,
```

Memory

Word | Cached

| Address | +0 | +4 | +8 | +12 | +16 | +20 |
|---|---|---|---|---|---|---|
| 0x000003d0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000003e8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000400 | 00000005 | 00000003 | 00000004 | 00000001 | 0000000f | 00000000 |
| 0x00000418 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000430 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000448 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000460 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000478 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000490 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004a8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004c0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |

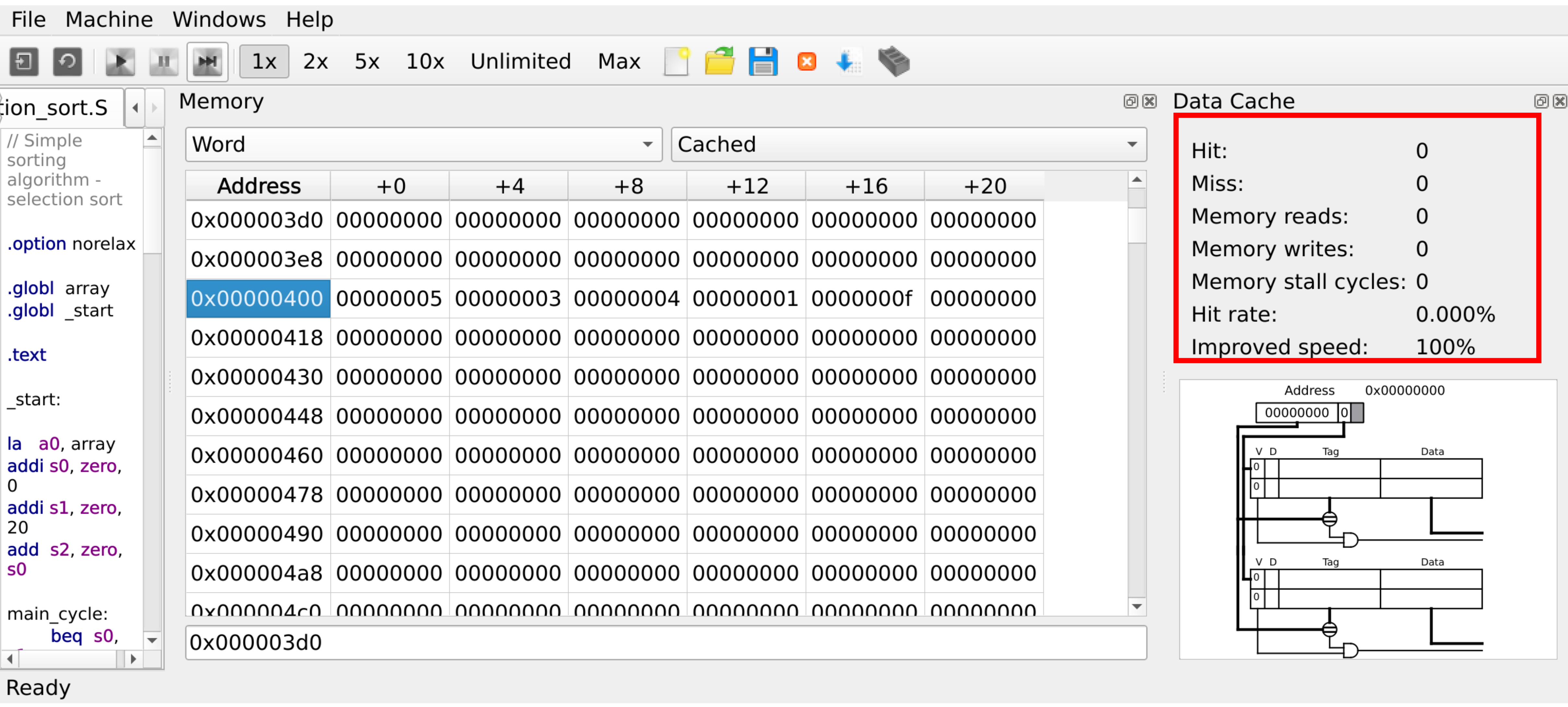
0x000003d0

Data Cache

| | |
|---|---|
| Hit: | 0 |
| Miss: | 0 |
| Memory reads: | 0 |
| Memory writes: | 0 |
| Memory stall cycles: | 0 |
| Hit rate: | 0.000% |
| Improved speed: | 100% |

Ready

# Cache View



```
// Simple
sorting
algorithm -
selection sort

.option norelax

.globl  array
.globl  _start

.text

_start:

la   a0, array
addi s0, zero,
0
addi s1, zero,
20
add  s2, zero,
s0

main_cycle:
        beq  s0,
```

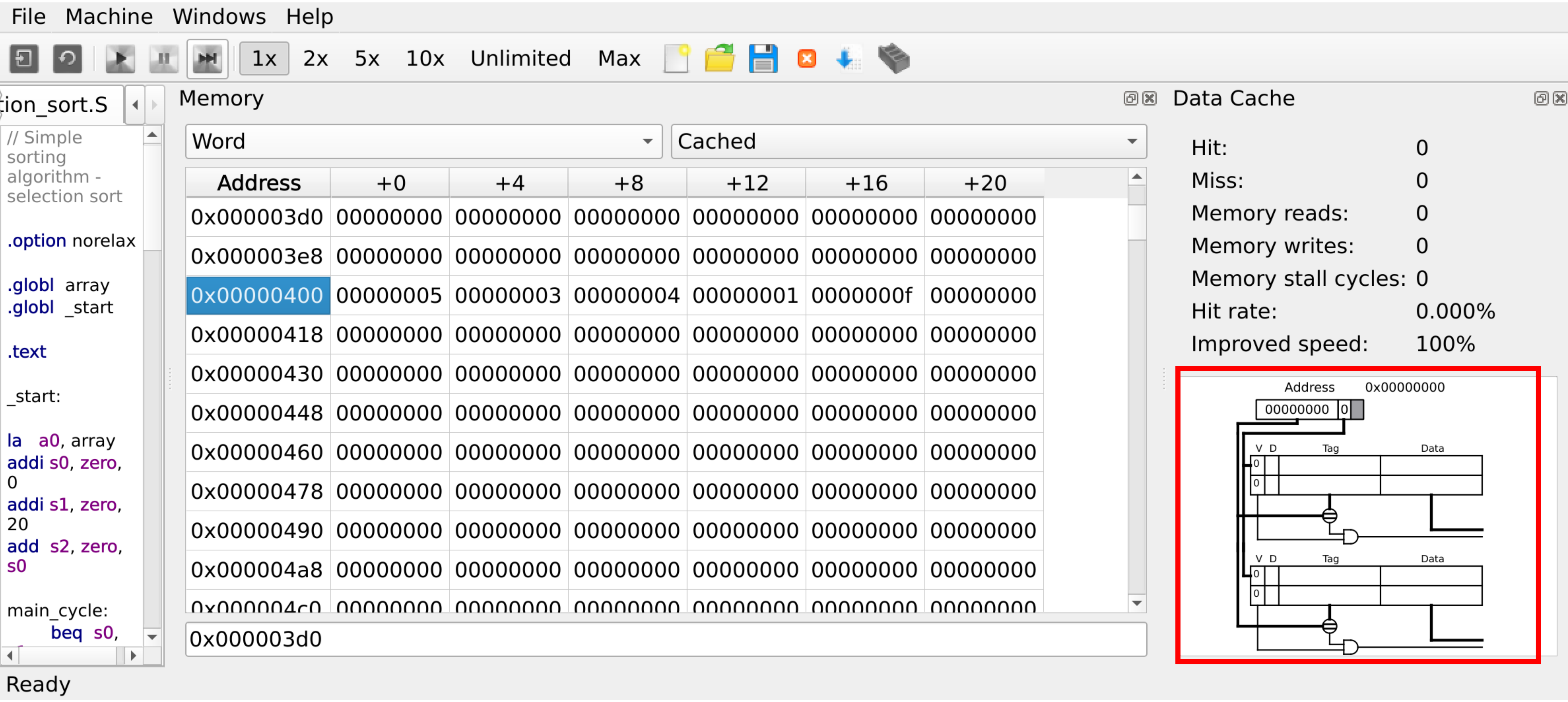
Word | Cached

| Address | +0 | +4 | +8 | +12 | +16 | +20 |
|---|---|---|---|---|---|---|
| 0x000003d0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000003e8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000400 | 00000005 | 00000003 | 00000004 | 00000001 | 0000000f | 00000000 |
| 0x00000418 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000430 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000448 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000460 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000478 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000490 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004a8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004c0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |

0x000003d0

| | |
|---|---|
| Hit: | 0 |
| Miss: | 0 |
| Memory reads: | 0 |
| Memory writes: | 0 |
| Memory stall cycles: | 0 |
| Hit rate: | 0.000% |
| Improved speed: | 100% |

# Cache View



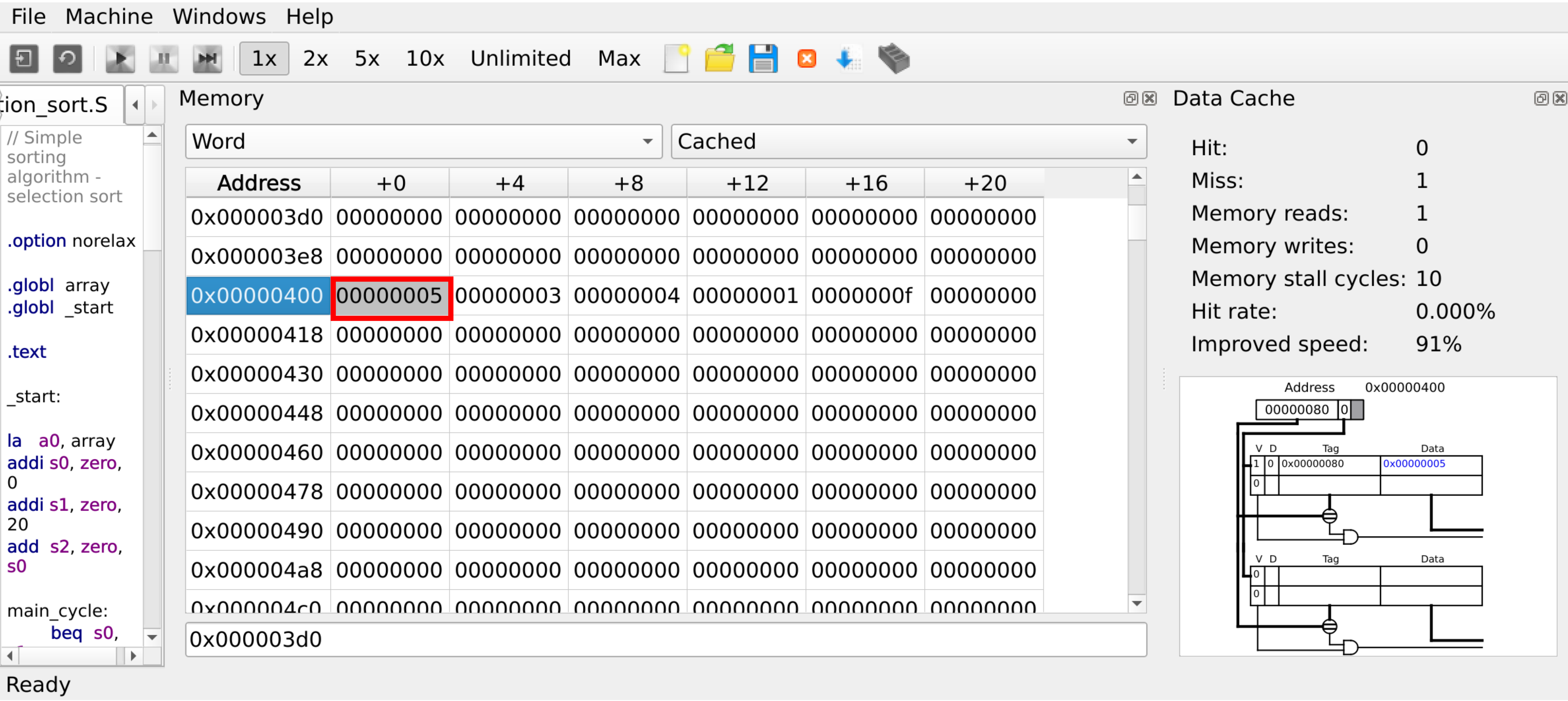
```
// Simple
sorting
algorithm -
selection sort

.option norelax

.globl  array
.globl  _start

.text

_start:

la   a0, array
addi s0, zero,
0
addi s1, zero,
20
add  s2, zero,
s0

main_cycle:
        beq  s0,
```

| Address | +0 | +4 | +8 | +12 | +16 | +20 |
|---|---|---|---|---|---|---|
| 0x000003d0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000003e8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000400 | 00000005 | 00000003 | 00000004 | 00000001 | 0000000f | 00000000 |
| 0x00000418 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000430 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000448 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000460 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000478 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000490 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004a8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004c0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |

0x000003d0

|  |  |
|---|---|
| Hit: | 0 |
| Miss: | 0 |
| Memory reads: | 0 |
| Memory writes: | 0 |
| Memory stall cycles: | 0 |
| Hit rate: | 0.000% |
| Improved speed: | 100% |

# Example: Selection Sort 1



```
// Simple
sorting
algorithm -
selection sort

.option norelax

.globl  array
.globl  _start

.text

_start:

la   a0, array
addi s0, zero,
0
addi s1, zero,
20
add  s2, zero,
s0

main_cycle:
        beq  s0,
```

Word | Cached

| Address | +0 | +4 | +8 | +12 | +16 | +20 |
|---|---|---|---|---|---|---|
| 0x000003d0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000003e8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000400 | 00000005 | 00000003 | 00000004 | 00000001 | 0000000f | 00000000 |
| 0x00000418 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000430 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000448 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000460 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000478 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000490 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004a8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004c0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |

0x000003d0

| | |
|---|---|
| Hit: | 0 |
| Miss: | 1 |
| Memory reads: | 1 |
| Memory writes: | 0 |
| Memory stall cycles: | 10 |
| Hit rate: | 0.000% |
| Improved speed: | 91% |

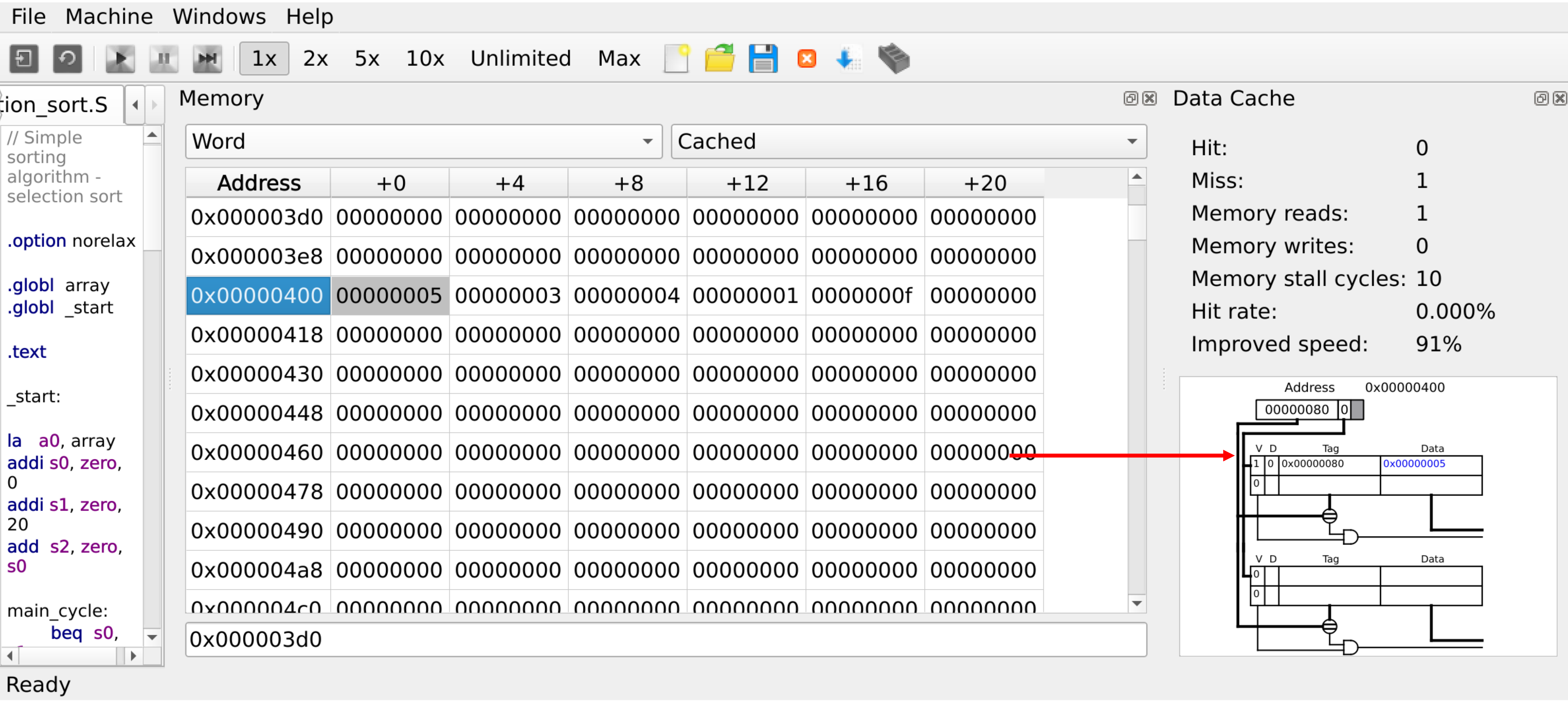
Address 0x00000400
00000080 | 0

| V | D | Tag | Data |
|---|---|---|---|
| 1 | 0 | 0x00000080 | 0x00000005 |
| 0 | | | |

| V | D | Tag | Data |
|---|---|---|---|
| 0 | | | |
| 0 | | | |

Ready

# Example: Selection Sort 1



```
// Simple
sorting
algorithm -
selection sort

.option norelax

.globl  array
.globl  _start

.text

_start:

la   a0, array
addi s0, zero,
0
addi s1, zero,
20
add  s2, zero,
s0

main_cycle:
     beq  s0,
```

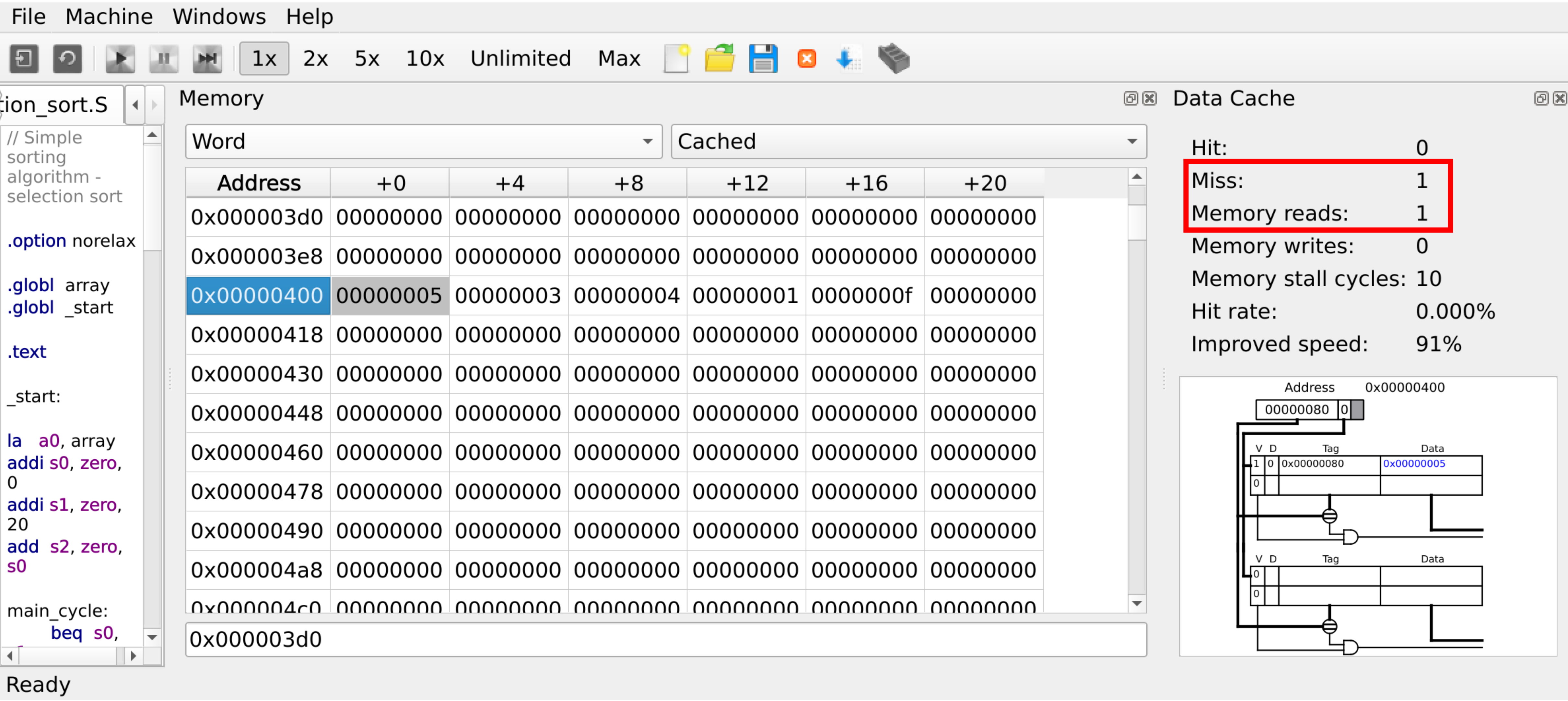
Word | Cached

| Address | +0 | +4 | +8 | +12 | +16 | +20 |
|---|---|---|---|---|---|---|
| 0x000003d0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000003e8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000400 | 00000005 | 00000003 | 00000004 | 00000001 | 0000000f | 00000000 |
| 0x00000418 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000430 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000448 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000460 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000478 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000490 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004a8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004c0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |

0x000003d0

| | |
|---|---|
| Hit: | 0 |
| Miss: | 1 |
| Memory reads: | 1 |
| Memory writes: | 0 |
| Memory stall cycles: | 10 |
| Hit rate: | 0.000% |
| Improved speed: | 91% |

# Example: Selection Sort 1



File Machine Windows Help

1x 2x 5x 10x Unlimited Max

```
ion_sort.S
// Simple
sorting
algorithm -
selection sort

.option norelax

.globl  array
.globl  _start

.text

_start:

la   a0, array
addi s0, zero,
0
addi s1, zero,
20
add  s2, zero,
s0

main_cycle:
     beq  s0,
```

Memory

Word · Cached

| Address | +0 | +4 | +8 | +12 | +16 | +20 |
|---|---|---|---|---|---|---|
| 0x000003d0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000003e8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000400 | 00000005 | 00000003 | 00000004 | 00000001 | 0000000f | 00000000 |
| 0x00000418 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000430 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000448 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000460 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000478 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000490 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004a8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004c0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |

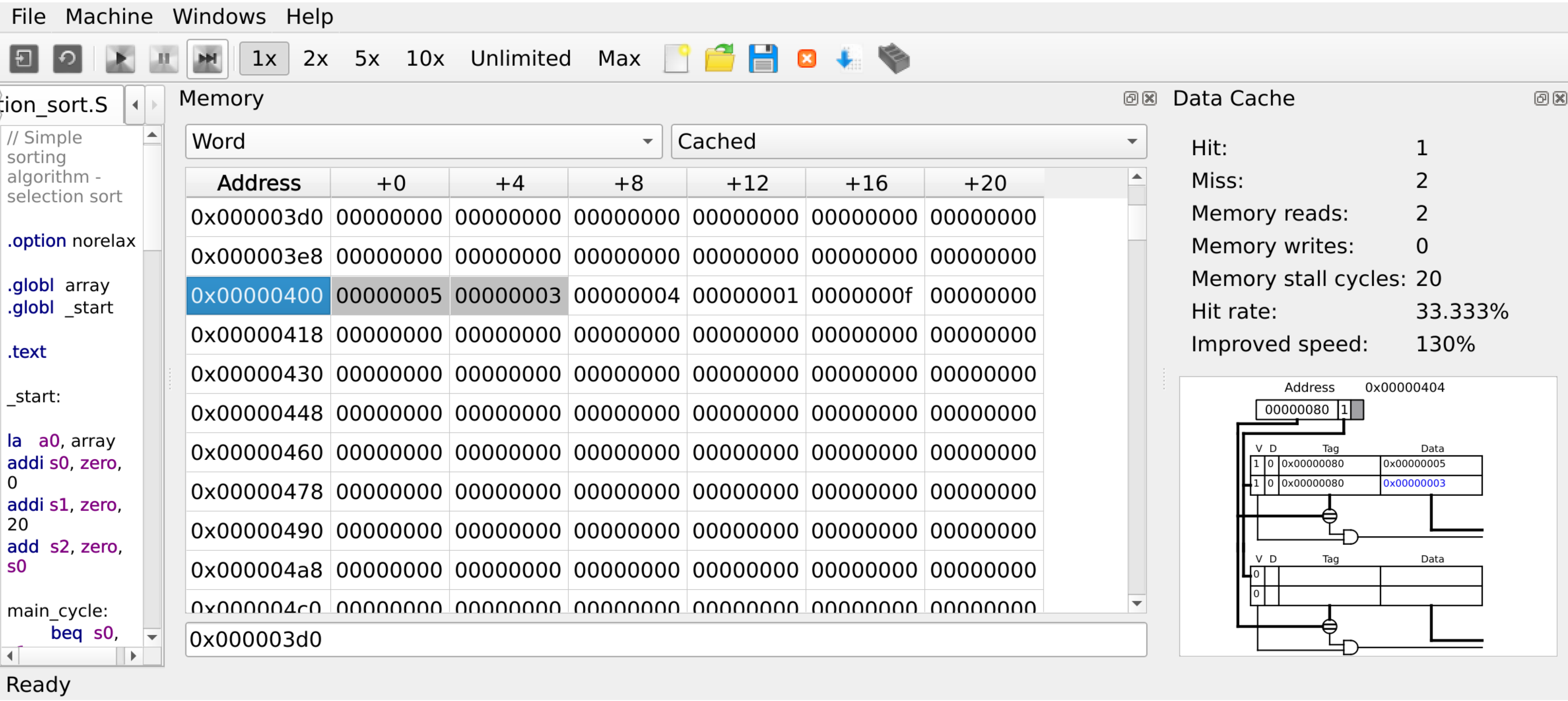
0x000003d0

Data Cache

| | |
|---|---|
| Hit: | 0 |
| Miss: | 1 |
| Memory reads: | 1 |
| Memory writes: | 0 |
| Memory stall cycles: | 10 |
| Hit rate: | 0.000% |
| Improved speed: | 91% |

Ready

# Example: Selection Sort 2



```
// Simple
sorting
algorithm -
selection sort

.option norelax

.globl  array
.globl  _start

.text

_start:

la   a0, array
addi s0, zero,
0
addi s1, zero,
20
add  s2, zero,
s0

main_cycle:
      beq  s0,
```

Word | Cached

| Address | +0 | +4 | +8 | +12 | +16 | +20 |
|---|---|---|---|---|---|---|
| 0x000003d0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000003e8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000400 | 00000005 | 00000003 | 00000004 | 00000001 | 0000000f | 00000000 |
| 0x00000418 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000430 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000448 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000460 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000478 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000490 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004a8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004c0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |

0x000003d0

| | |
|---|---|
| Hit: | 1 |
| Miss: | 2 |
| Memory reads: | 2 |
| Memory writes: | 0 |
| Memory stall cycles: | 20 |
| Hit rate: | 33.333% |
| Improved speed: | 130% |

Address 0x00000404

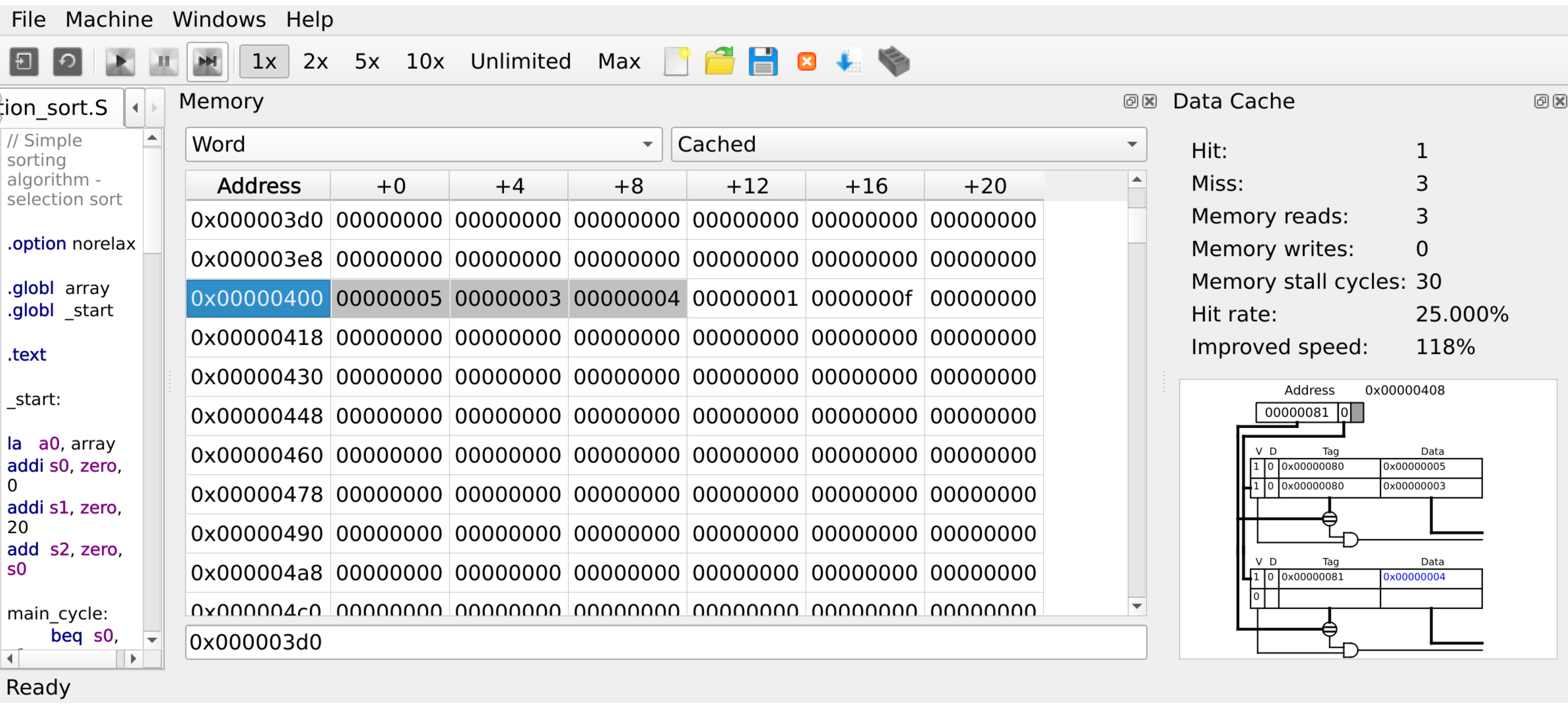
00000080 | 1

| V | D | Tag | Data |
|---|---|---|---|
| 1 | 0 | 0x00000080 | 0x00000005 |
| 1 | 0 | 0x00000080 | 0x00000003 |

| V | D | Tag | Data |
|---|---|---|---|
| 0 | | | |
| 0 | | | |

Ready

# Example: Selection Sort 3



```
ion_sort.S
// Simple
sorting
algorithm -
selection sort

.option norelax

.globl  array
.globl  _start

.text

_start:

la   a0, array
addi s0, zero,
0
addi s1, zero,
20
add  s2, zero,
s0

main_cycle:
     beq  s0,
```

| Address | +0 | +4 | +8 | +12 | +16 | +20 |
|---|---|---|---|---|---|---|
| 0x000003d0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000003e8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000400 | 00000005 | 00000003 | 00000004 | 00000001 | 0000000f | 00000000 |
| 0x00000418 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000430 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000448 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000460 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000478 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000490 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004a8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004c0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |

0x000003d0

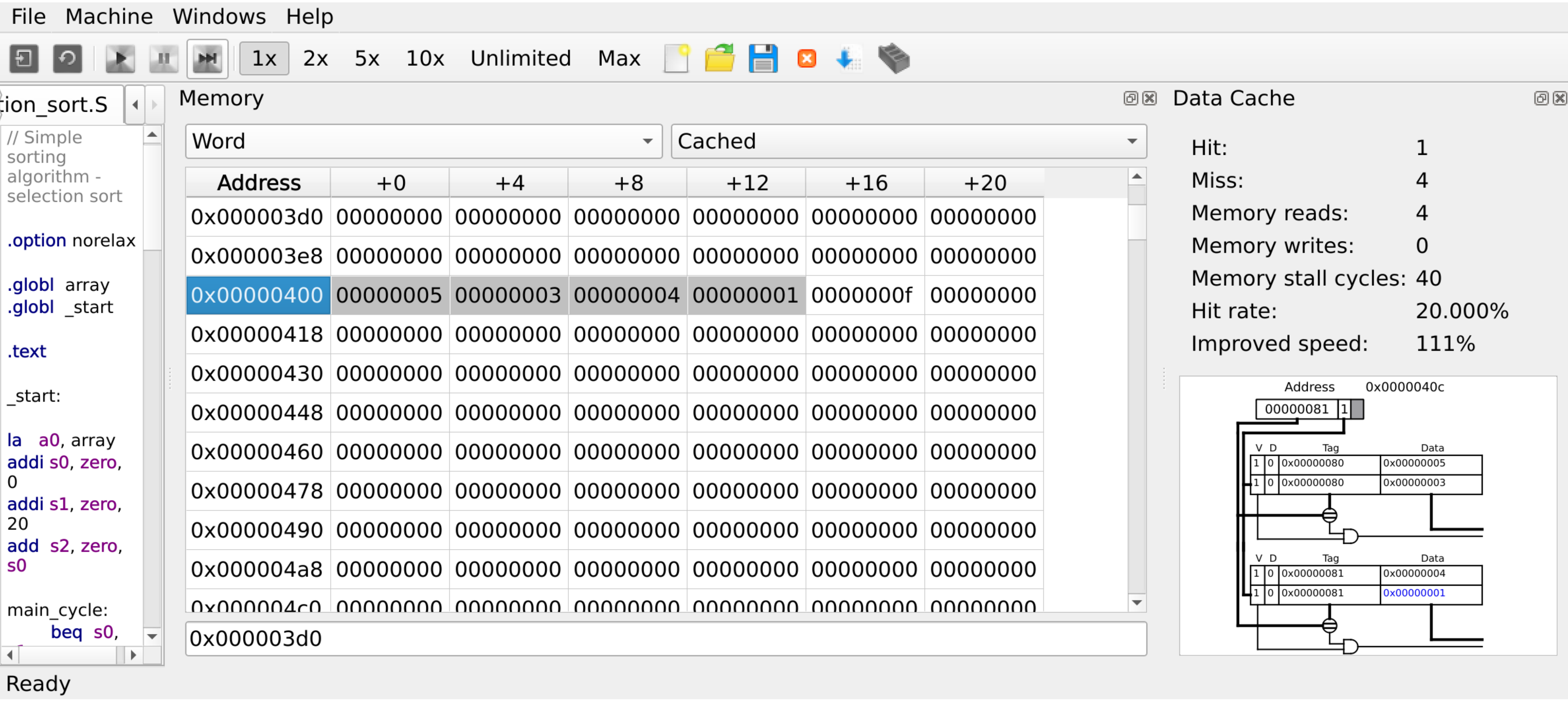
| | |
|---|---|
| Hit: | 1 |
| Miss: | 3 |
| Memory reads: | 3 |
| Memory writes: | 0 |
| Memory stall cycles: | 30 |
| Hit rate: | 25.000% |
| Improved speed: | 118% |

Address 0x00000408
00000081 | 0

| V | D | Tag | Data |
|---|---|---|---|
| 1 | 0 | 0x00000080 | 0x00000005 |
| 1 | 0 | 0x00000080 | 0x00000003 |

| V | D | Tag | Data |
|---|---|---|---|
| 1 | 0 | 0x00000081 | 0x00000004 |
| 0 | | | |

# Example: Selection Sort 4



```
// Simple
sorting
algorithm -
selection sort

.option norelax

.globl  array
.globl  _start

.text

_start:

la   a0, array
addi s0, zero,
0
addi s1, zero,
20
add  s2, zero,
s0

main_cycle:
      beq  s0,
```

| Address | +0 | +4 | +8 | +12 | +16 | +20 |
|---|---|---|---|---|---|---|
| 0x000003d0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000003e8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000400 | 00000005 | 00000003 | 00000004 | 00000001 | 0000000f | 00000000 |
| 0x00000418 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000430 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000448 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000460 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000478 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000490 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004a8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004c0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |

0x000003d0

| | |
|---|---|
| Hit: | 1 |
| Miss: | 4 |
| Memory reads: | 4 |
| Memory writes: | 0 |
| Memory stall cycles: | 40 |
| Hit rate: | 20.000% |
| Improved speed: | 111% |

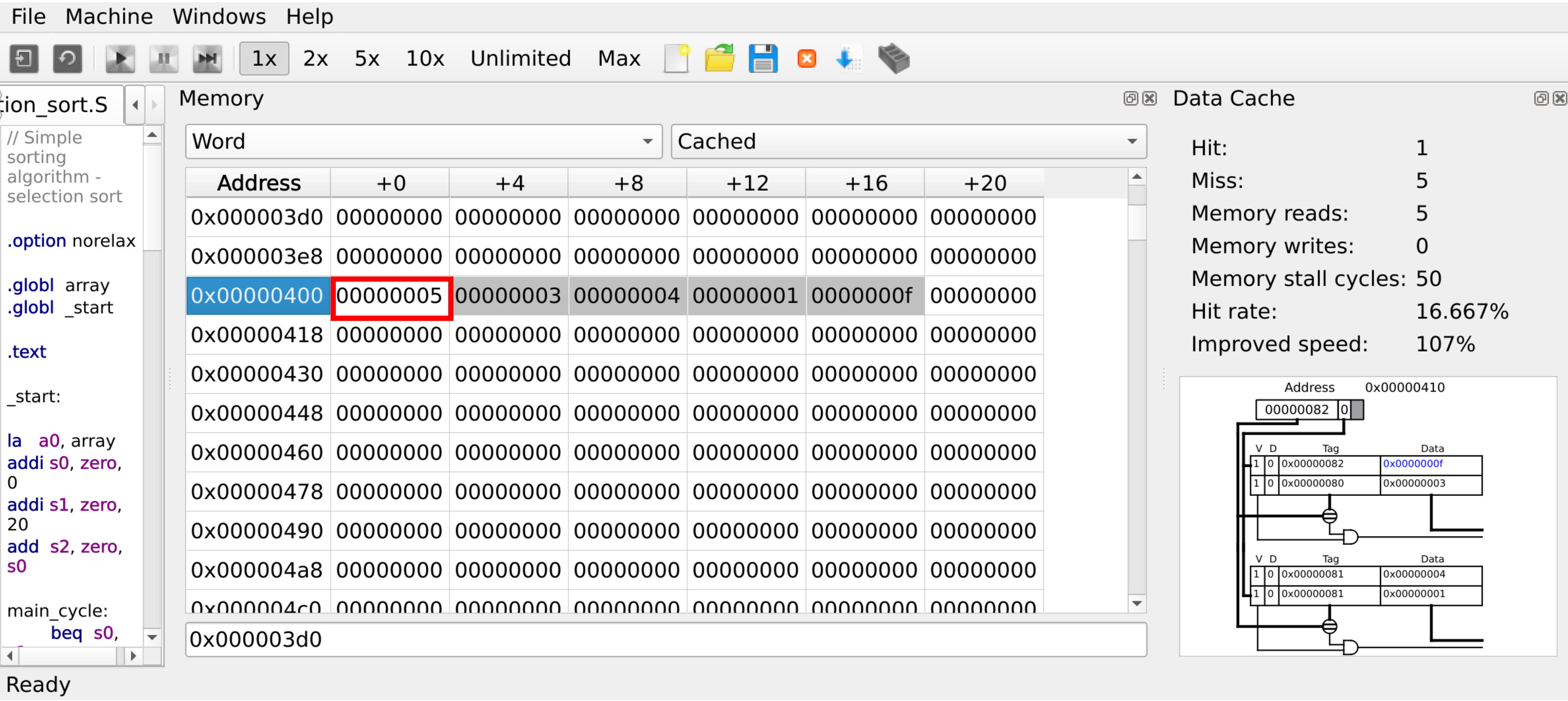
Address 0x0000040c

00000081 | 1

| V | D | Tag | Data |
|---|---|---|---|
| 1 | 0 | 0x00000080 | 0x00000005 |
| 1 | 0 | 0x00000080 | 0x00000003 |

| V | D | Tag | Data |
|---|---|---|---|
| 1 | 0 | 0x00000081 | 0x00000004 |
| 1 | 0 | 0x00000081 | 0x00000001 |

# Example: Selection Sort 5



```
// Simple
sorting
algorithm -
selection sort

.option norelax

.globl  array
.globl  _start

.text

_start:

la   a0, array
addi s0, zero,
0
addi s1, zero,
20
add  s2, zero,
s0

main_cycle:
        beq  s0,
```

Word — Cached

| Address | +0 | +4 | +8 | +12 | +16 | +20 |
|---|---|---|---|---|---|---|
| 0x000003d0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000003e8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000400 | 00000005 | 00000003 | 00000004 | 00000001 | 0000000f | 00000000 |
| 0x00000418 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000430 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000448 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000460 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000478 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000490 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004a8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004c0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |

0x000003d0

| | |
|---|---|
| Hit: | 1 |
| Miss: | 5 |
| Memory reads: | 5 |
| Memory writes: | 0 |
| Memory stall cycles: | 50 |
| Hit rate: | 16.667% |
| Improved speed: | 107% |

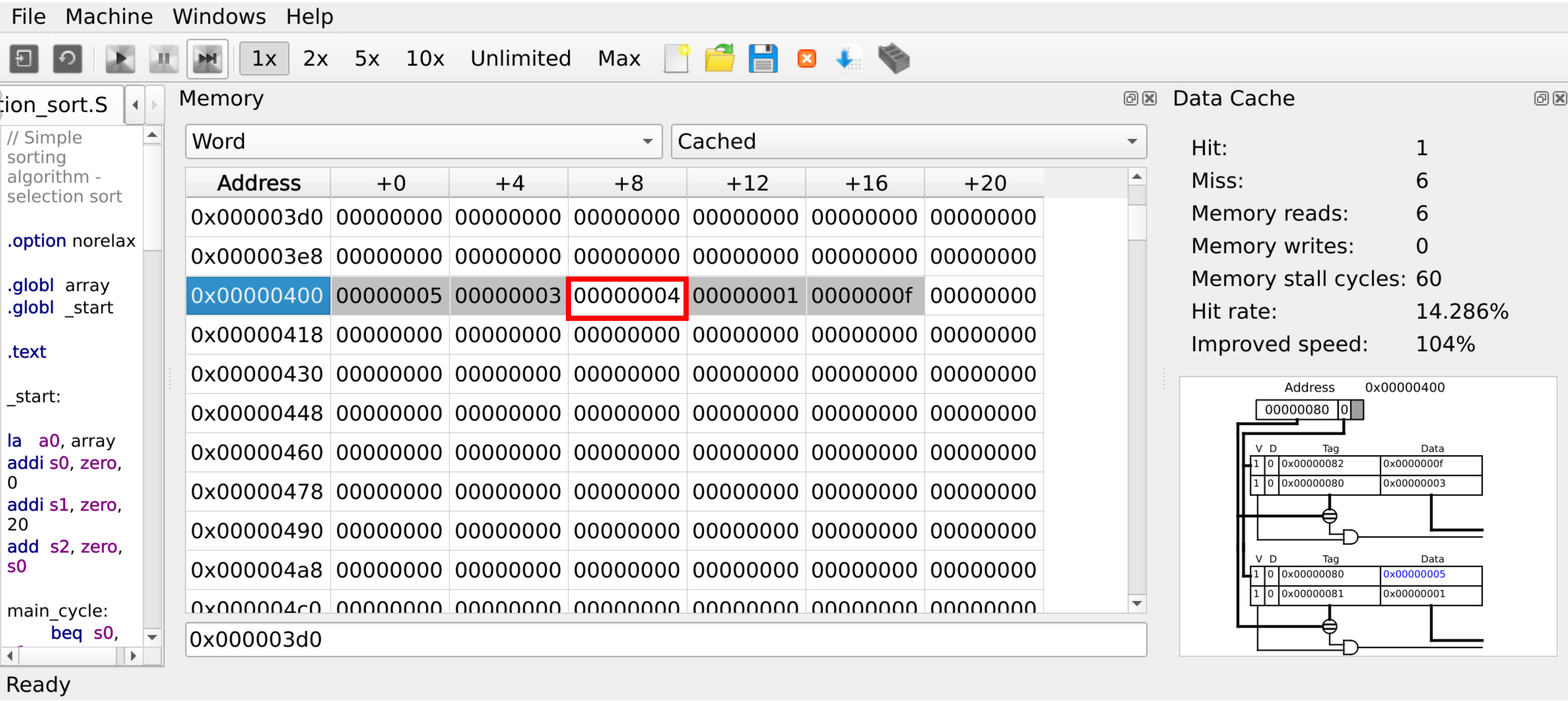
Address 0x00000410

00000082 | 0

| V | D | Tag | Data |
|---|---|---|---|
| 1 | 0 | 0x00000082 | 0x0000000f |
| 1 | 0 | 0x00000080 | 0x00000003 |

| V | D | Tag | Data |
|---|---|---|---|
| 1 | 0 | 0x00000081 | 0x00000004 |
| 1 | 0 | 0x00000081 | 0x00000001 |

# Example: Selection Sort 6



```
// Simple
sorting
algorithm -
selection sort

.option norelax

.globl  array
.globl  _start

.text

_start:

la   a0, array
addi s0, zero,
0
addi s1, zero,
20
add  s2, zero,
s0

main_cycle:
     beq  s0,
```

Word | Cached

| Address | +0 | +4 | +8 | +12 | +16 | +20 |
|---|---|---|---|---|---|---|
| 0x000003d0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000003e8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000400 | 00000005 | 00000003 | 00000004 | 00000001 | 0000000f | 00000000 |
| 0x00000418 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000430 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000448 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000460 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000478 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000490 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004a8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004c0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |

0x000003d0

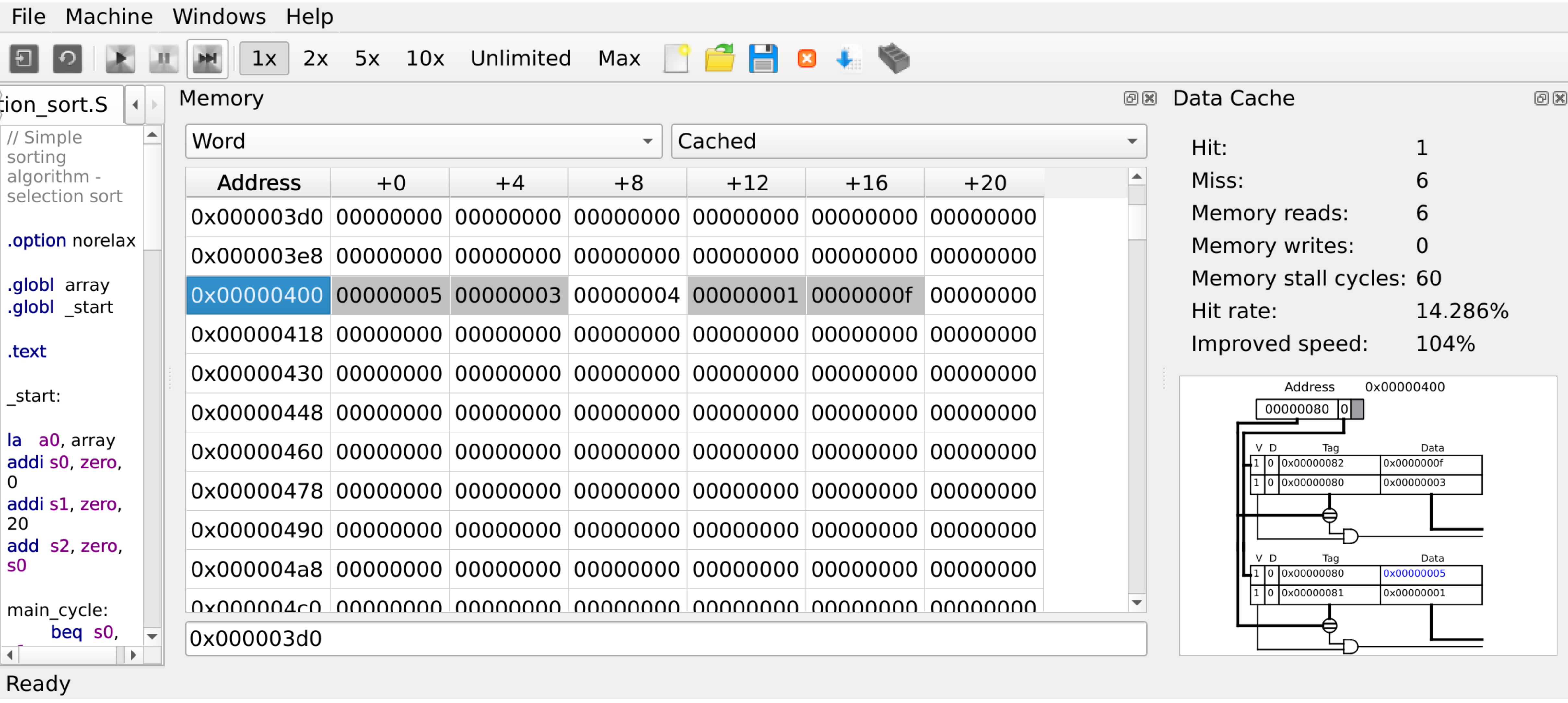
| | |
|---|---|
| Hit: | 1 |
| Miss: | 6 |
| Memory reads: | 6 |
| Memory writes: | 0 |
| Memory stall cycles: | 60 |
| Hit rate: | 14.286% |
| Improved speed: | 104% |

Address 0x00000400
00000080 | 0

| V | D | Tag | Data |
|---|---|---|---|
| 1 | 0 | 0x00000082 | 0x0000000f |
| 1 | 0 | 0x00000080 | 0x00000003 |

| V | D | Tag | Data |
|---|---|---|---|
| 1 | 0 | 0x00000080 | 0x00000005 |
| 1 | 0 | 0x00000081 | 0x00000001 |

# Example: Selection Sort 6



```
// Simple
sorting
algorithm -
selection sort

.option norelax

.globl  array
.globl  _start

.text

_start:

la   a0, array
addi s0, zero,
0
addi s1, zero,
20
add  s2, zero,
s0

main_cycle:
     beq  s0,
```

| Address | +0 | +4 | +8 | +12 | +16 | +20 |
|---|---|---|---|---|---|---|
| 0x000003d0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000003e8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000400 | 00000005 | 00000003 | 00000004 | 00000001 | 0000000f | 00000000 |
| 0x00000418 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000430 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000448 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000460 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000478 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000490 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004a8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004c0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |

0x000003d0

| | |
|---|---|
| Hit: | 1 |
| Miss: | 6 |
| Memory reads: | 6 |
| Memory writes: | 0 |
| Memory stall cycles: | 60 |
| Hit rate: | 14.286% |
| Improved speed: | 104% |

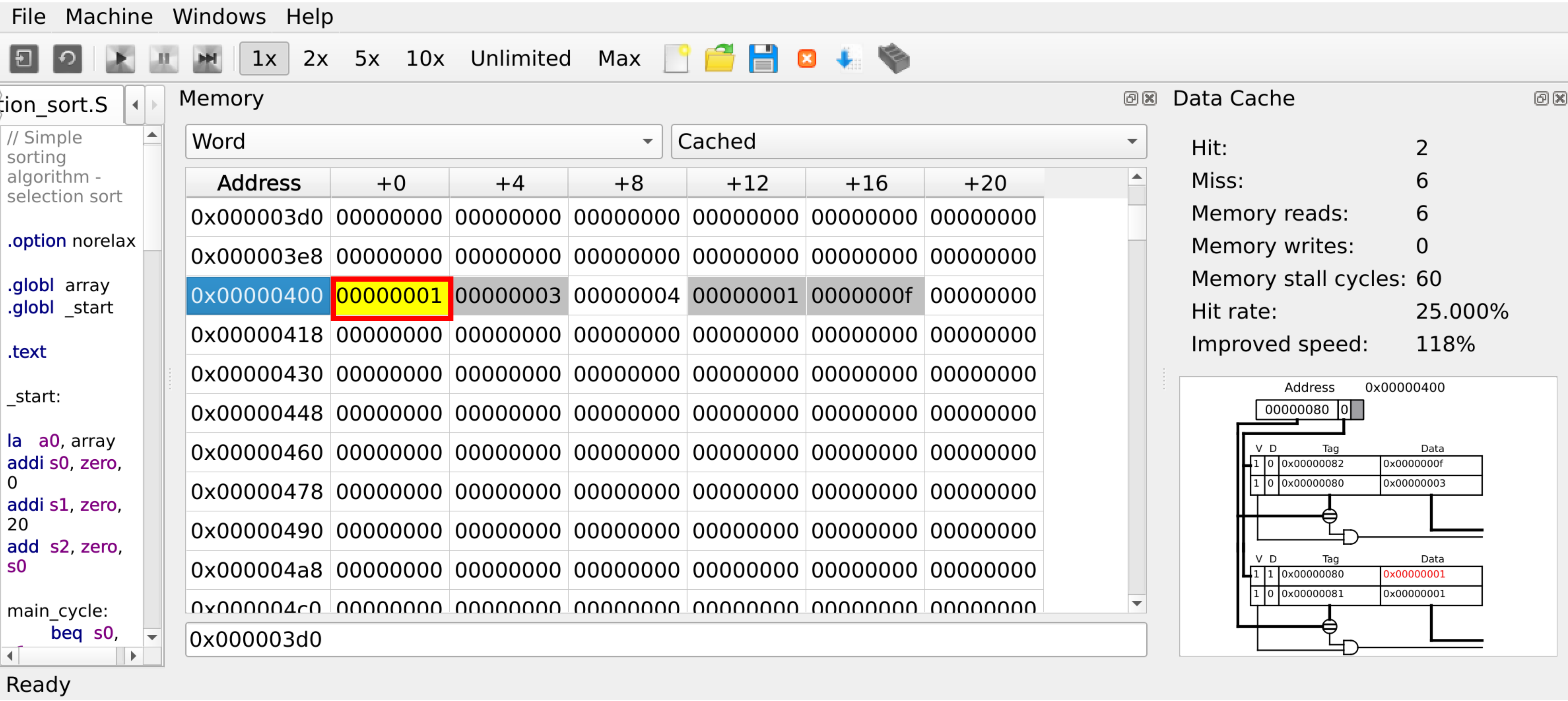
Address 0x00000400

00000080 | 0

| V | D | Tag | Data |
|---|---|---|---|
| 1 | 0 | 0x00000082 | 0x0000000f |
| 1 | 0 | 0x00000080 | 0x00000003 |

| V | D | Tag | Data |
|---|---|---|---|
| 1 | 0 | 0x00000080 | 0x00000005 |
| 1 | 0 | 0x00000081 | 0x00000001 |

# Example: Selection Sort 7



```
// Simple
sorting
algorithm -
selection sort

.option norelax

.globl  array
.globl  _start

.text

_start:

la   a0, array
addi s0, zero,
0
addi s1, zero,
20
add  s2, zero,
s0

main_cycle:
      beq  s0,
```

| Address | +0 | +4 | +8 | +12 | +16 | +20 |
|---|---|---|---|---|---|---|
| 0x000003d0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000003e8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000400 | 00000001 | 00000003 | 00000004 | 00000001 | 0000000f | 00000000 |
| 0x00000418 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000430 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000448 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000460 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000478 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000490 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004a8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004c0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |

0x000003d0

| | |
|---|---|
| Hit: | 2 |
| Miss: | 6 |
| Memory reads: | 6 |
| Memory writes: | 0 |
| Memory stall cycles: | 60 |
| Hit rate: | 25.000% |
| Improved speed: | 118% |

Address 0x00000400

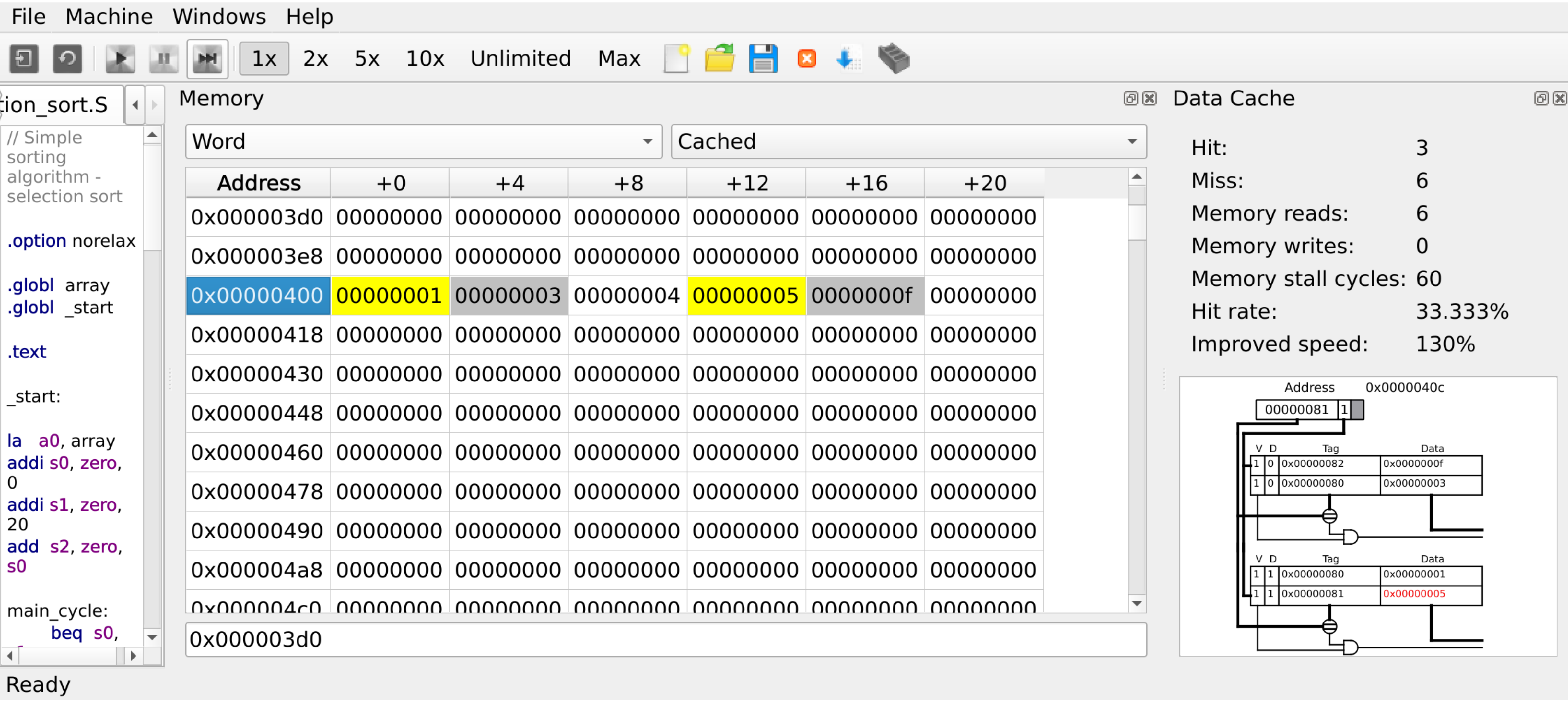
00000080 | 0

| V | D | Tag | Data |
|---|---|---|---|
| 1 | 0 | 0x00000082 | 0x0000000f |
| 1 | 0 | 0x00000080 | 0x00000003 |

| V | D | Tag | Data |
|---|---|---|---|
| 1 | 1 | 0x00000080 | 0x00000001 |
| 1 | 0 | 0x00000081 | 0x00000001 |

Ready

# Example: Selection Sort 8



File Machine Windows Help

1x 2x 5x 10x Unlimited Max

```
ion_sort.S

// Simple
sorting
algorithm -
selection sort

.option norelax

.globl  array
.globl  _start

.text

_start:

la   a0, array
addi s0, zero,
0
addi s1, zero,
20
add  s2, zero,
s0

main_cycle:
     beq  s0,
```

Memory

Word | Cached

| Address | +0 | +4 | +8 | +12 | +16 | +20 |
|---|---|---|---|---|---|---|
| 0x000003d0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000003e8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000400 | 00000001 | 00000003 | 00000004 | 00000005 | 0000000f | 00000000 |
| 0x00000418 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000430 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000448 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000460 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000478 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000490 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004a8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004c0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |

0x000003d0

Data Cache

| | |
|---|---|
| Hit: | 3 |
| Miss: | 6 |
| Memory reads: | 6 |
| Memory writes: | 0 |
| Memory stall cycles: | 60 |
| Hit rate: | 33.333% |
| Improved speed: | 130% |

Address 0x0000040c

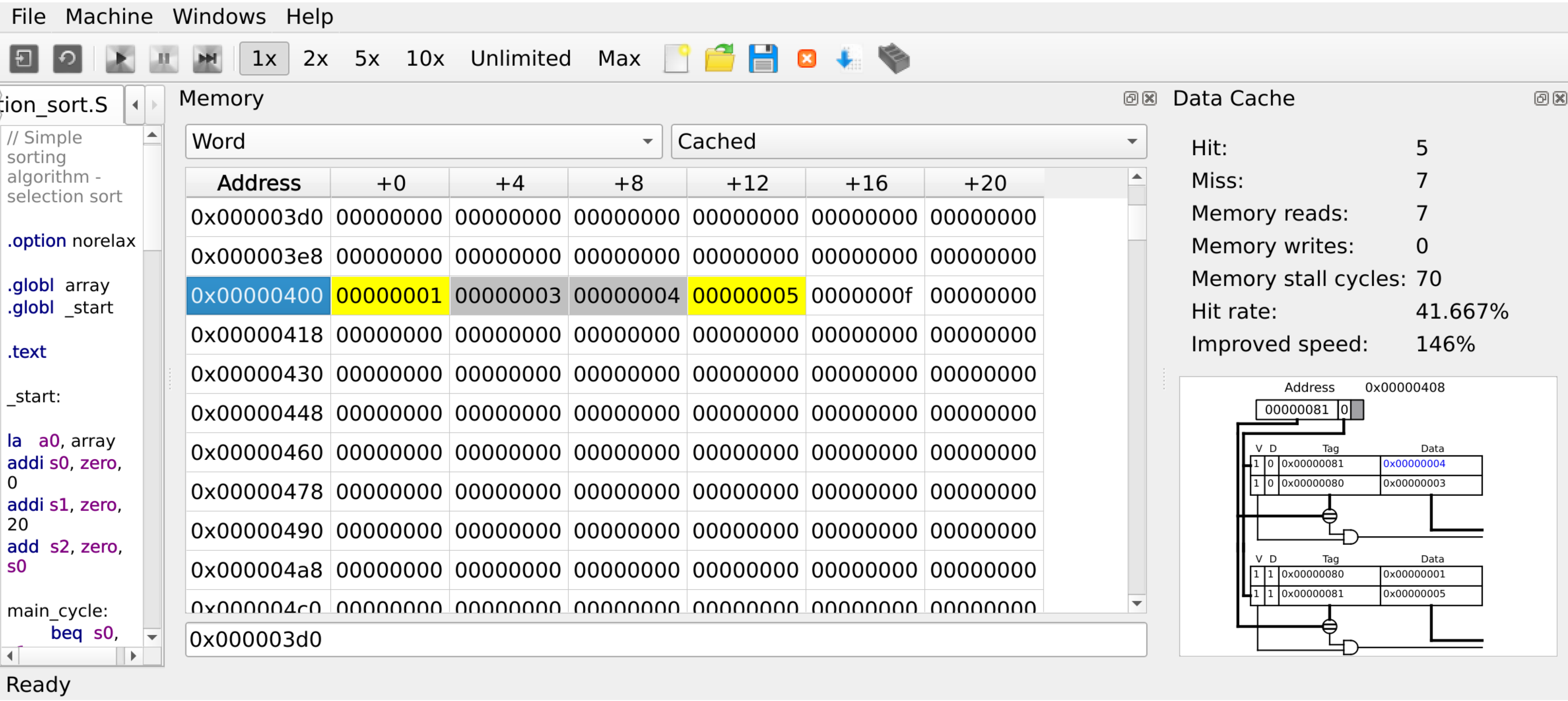
00000081 | 1

| V | D | Tag | Data |
|---|---|---|---|
| 1 | 0 | 0x00000082 | 0x0000000f |
| 1 | 0 | 0x00000080 | 0x00000003 |

| V | D | Tag | Data |
|---|---|---|---|
| 1 | 1 | 0x00000080 | 0x00000001 |
| 1 | 1 | 0x00000081 | 0x00000005 |

Ready

# Example: Selection Sort 9



```
// Simple
sorting
algorithm -
selection sort

.option norelax

.globl  array
.globl  _start

.text

_start:

la   a0, array
addi s0, zero,
0
addi s1, zero,
20
add  s2, zero,
s0

main_cycle:
        beq  s0,
```

Word | Cached

| Address | +0 | +4 | +8 | +12 | +16 | +20 |
|---|---|---|---|---|---|---|
| 0x000003d0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000003e8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000400 | 00000001 | 00000003 | 00000004 | 00000005 | 0000000f | 00000000 |
| 0x00000418 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000430 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000448 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000460 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000478 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000490 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004a8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004c0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |

0x000003d0

| | |
|---|---|
| Hit: | 5 |
| Miss: | 7 |
| Memory reads: | 7 |
| Memory writes: | 0 |
| Memory stall cycles: | 70 |
| Hit rate: | 41.667% |
| Improved speed: | 146% |

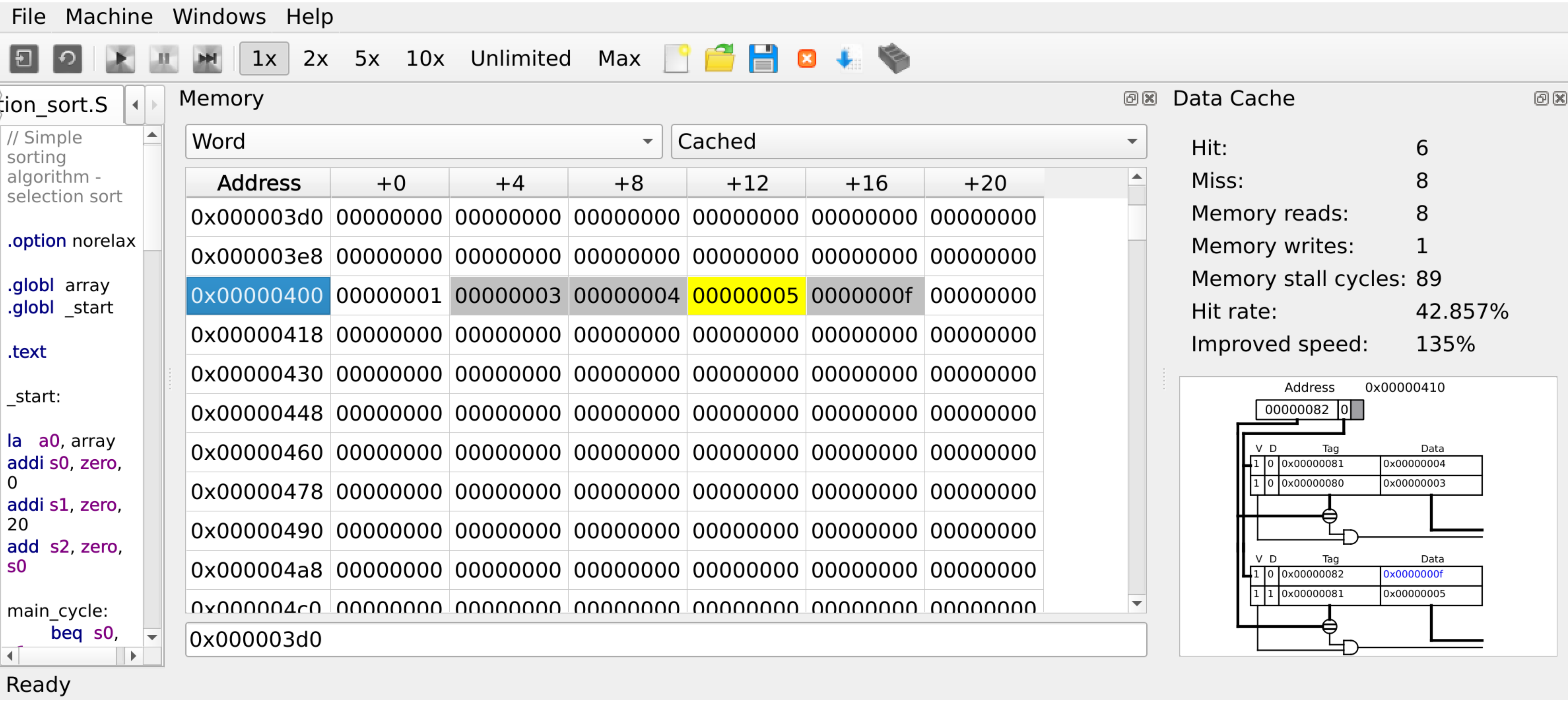
Address 0x00000408

00000081 | 0

| V | D | Tag | Data |
|---|---|---|---|
| 1 | 0 | 0x00000081 | 0x00000004 |
| 1 | 0 | 0x00000080 | 0x00000003 |

| V | D | Tag | Data |
|---|---|---|---|
| 1 | 1 | 0x00000080 | 0x00000001 |
| 1 | 1 | 0x00000081 | 0x00000005 |

# Example: Selection Sort 10



```
// Simple
sorting
algorithm -
selection sort

.option norelax

.globl  array
.globl  _start

.text

_start:

la   a0, array
addi s0, zero,
0
addi s1, zero,
20
add  s2, zero,
s0

main_cycle:
        beq  s0,
```

| Address | +0 | +4 | +8 | +12 | +16 | +20 |
|---|---|---|---|---|---|---|
| 0x000003d0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000003e8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000400 | 00000001 | 00000003 | 00000004 | 00000005 | 0000000f | 00000000 |
| 0x00000418 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000430 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000448 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000460 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000478 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x00000490 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004a8 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| 0x000004c0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |

0x000003d0

| | |
|---|---|
| Hit: | 6 |
| Miss: | 8 |
| Memory reads: | 8 |
| Memory writes: | 1 |
| Memory stall cycles: | 89 |
| Hit rate: | 42.857% |
| Improved speed: | 135% |

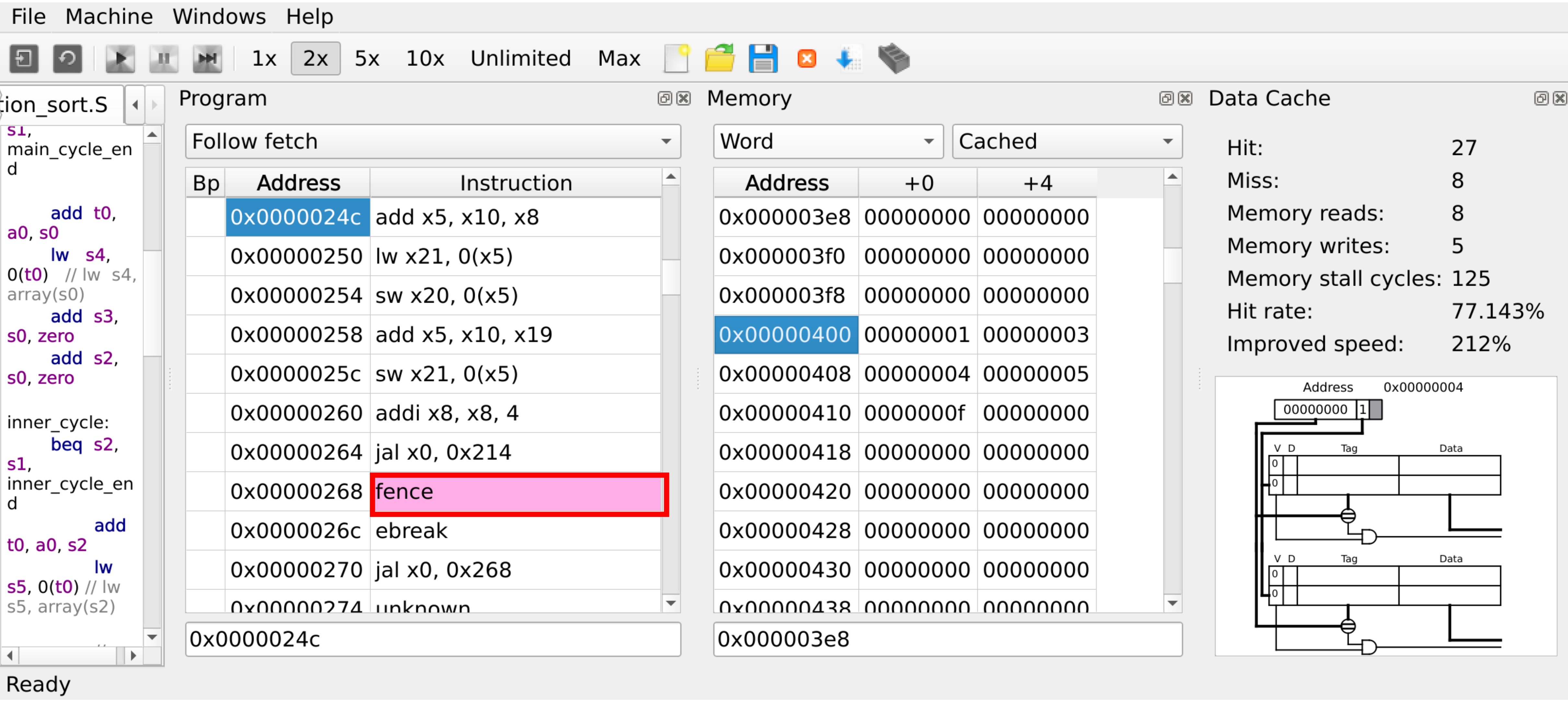
Address 0x00000410

00000082 | 0

| V | D | Tag | Data |
|---|---|---|---|
| 1 | 0 | 0x00000081 | 0x00000004 |
| 1 | 0 | 0x00000080 | 0x00000003 |

| V | D | Tag | Data |
|---|---|---|---|
| 1 | 0 | 0x00000082 | 0x0000000f |
| 1 | 1 | 0x00000081 | 0x00000005 |

# Example: Selection Sort 11 - Fence



File Machine Windows Help

1x 2x 5x 10x Unlimited Max

```
ion_sort.S
s1,
main_cycle_en
d

        add  t0,
a0, s0
        lw   s4,
0(t0)   // lw  s4,
array(s0)
        add  s3,
s0, zero
        add  s2,
s0, zero

inner_cycle:
        beq  s2,
s1,
inner_cycle_en
d
             add
t0, a0, s2
             lw
s5, 0(t0) // lw
s5, array(s2)
```

Program

Follow fetch

| Bp | Address | Instruction |
|---|---|---|
| | 0x0000024c | add x5, x10, x8 |
| | 0x00000250 | lw x21, 0(x5) |
| | 0x00000254 | sw x20, 0(x5) |
| | 0x00000258 | add x5, x10, x19 |
| | 0x0000025c | sw x21, 0(x5) |
| | 0x00000260 | addi x8, x8, 4 |
| | 0x00000264 | jal x0, 0x214 |
| | 0x00000268 | fence |
| | 0x0000026c | ebreak |
| | 0x00000270 | jal x0, 0x268 |
| | 0x00000274 | unknown |

0x0000024c

Memory

Word | Cached

| Address | +0 | +4 |
|---|---|---|
| 0x000003e8 | 00000000 | 00000000 |
| 0x000003f0 | 00000000 | 00000000 |
| 0x000003f8 | 00000000 | 00000000 |
| 0x00000400 | 00000001 | 00000003 |
| 0x00000408 | 00000004 | 00000005 |
| 0x00000410 | 0000000f | 00000000 |
| 0x00000418 | 00000000 | 00000000 |
| 0x00000420 | 00000000 | 00000000 |
| 0x00000428 | 00000000 | 00000000 |
| 0x00000430 | 00000000 | 00000000 |
| 0x00000438 | 00000000 | 00000000 |

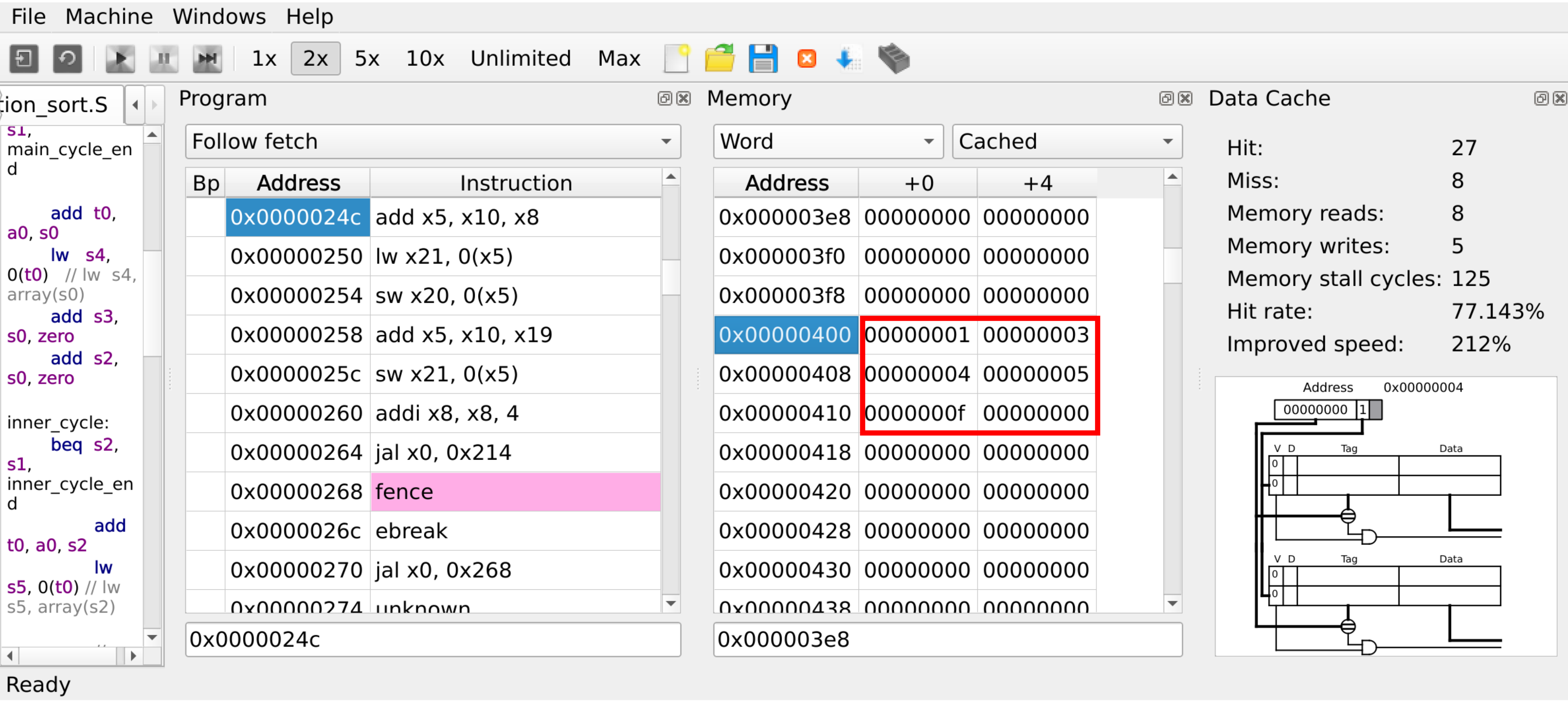
0x000003e8

Data Cache

| | |
|---|---|
| Hit: | 27 |
| Miss: | 8 |
| Memory reads: | 8 |
| Memory writes: | 5 |
| Memory stall cycles: | 125 |
| Hit rate: | 77.143% |
| Improved speed: | 212% |

Ready

# Example: Selection Sort 11 - Fence



```
ion_sort.S
s1,
main_cycle_en
d

        add  t0,
a0, s0
        lw   s4,
0(t0)   // lw  s4,
array(s0)
        add  s3,
s0, zero
        add  s2,
s0, zero

inner_cycle:
        beq  s2,
s1,
inner_cycle_en
d
                add
t0, a0, s2
                lw
s5, 0(t0) // lw
s5, array(s2)
```

| Bp | Address | Instruction |
|---|---|---|
| | 0x0000024c | add x5, x10, x8 |
| | 0x00000250 | lw x21, 0(x5) |
| | 0x00000254 | sw x20, 0(x5) |
| | 0x00000258 | add x5, x10, x19 |
| | 0x0000025c | sw x21, 0(x5) |
| | 0x00000260 | addi x8, x8, 4 |
| | 0x00000264 | jal x0, 0x214 |
| | 0x00000268 | fence |
| | 0x0000026c | ebreak |
| | 0x00000270 | jal x0, 0x268 |
| | 0x00000274 | unknown |

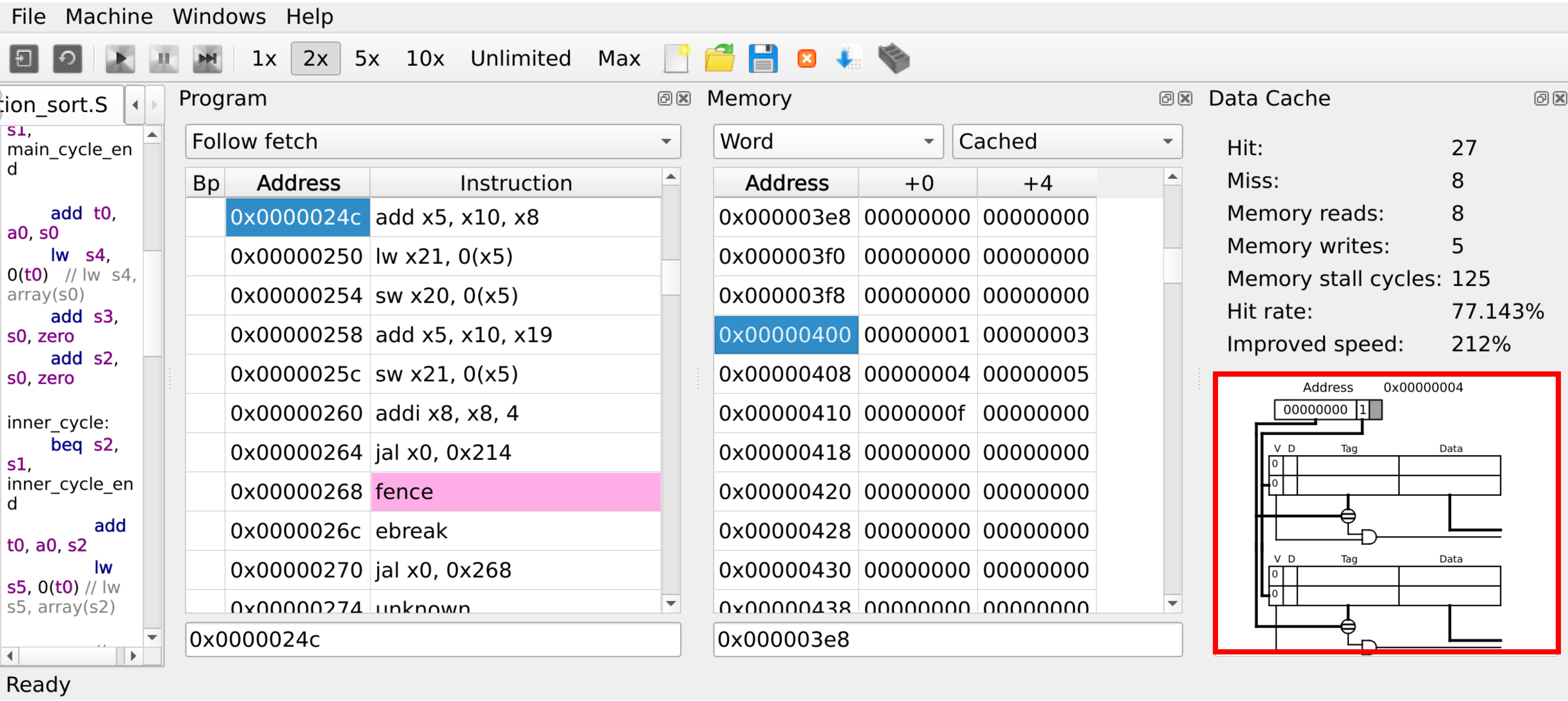
0x0000024c

| Address | +0 | +4 |
|---|---|---|
| 0x000003e8 | 00000000 | 00000000 |
| 0x000003f0 | 00000000 | 00000000 |
| 0x000003f8 | 00000000 | 00000000 |
| 0x00000400 | 00000001 | 00000003 |
| 0x00000408 | 00000004 | 00000005 |
| 0x00000410 | 0000000f | 00000000 |
| 0x00000418 | 00000000 | 00000000 |
| 0x00000420 | 00000000 | 00000000 |
| 0x00000428 | 00000000 | 00000000 |
| 0x00000430 | 00000000 | 00000000 |
| 0x00000438 | 00000000 | 00000000 |

0x000003e8

| | |
|---|---|
| Hit: | 27 |
| Miss: | 8 |
| Memory reads: | 8 |
| Memory writes: | 5 |
| Memory stall cycles: | 125 |
| Hit rate: | 77.143% |
| Improved speed: | 212% |

# Example: Selection Sort 11 - Fence



```
s1,
main_cycle_en
d

        add  t0,
a0, s0
        lw   s4,
0(t0)   // lw  s4,
array(s0)
        add  s3,
s0, zero
        add  s2,
s0, zero

inner_cycle:
        beq  s2,
s1,
inner_cycle_en
d
            add
t0, a0, s2
            lw
s5, 0(t0) // lw
s5, array(s2)
```

Program — Follow fetch

| Bp | Address | Instruction |
|---|---|---|
| | 0x0000024c | add x5, x10, x8 |
| | 0x00000250 | lw x21, 0(x5) |
| | 0x00000254 | sw x20, 0(x5) |
| | 0x00000258 | add x5, x10, x19 |
| | 0x0000025c | sw x21, 0(x5) |
| | 0x00000260 | addi x8, x8, 4 |
| | 0x00000264 | jal x0, 0x214 |
| | 0x00000268 | fence |
| | 0x0000026c | ebreak |
| | 0x00000270 | jal x0, 0x268 |
| | 0x00000274 | unknown |

0x0000024c

Memory — Word — Cached

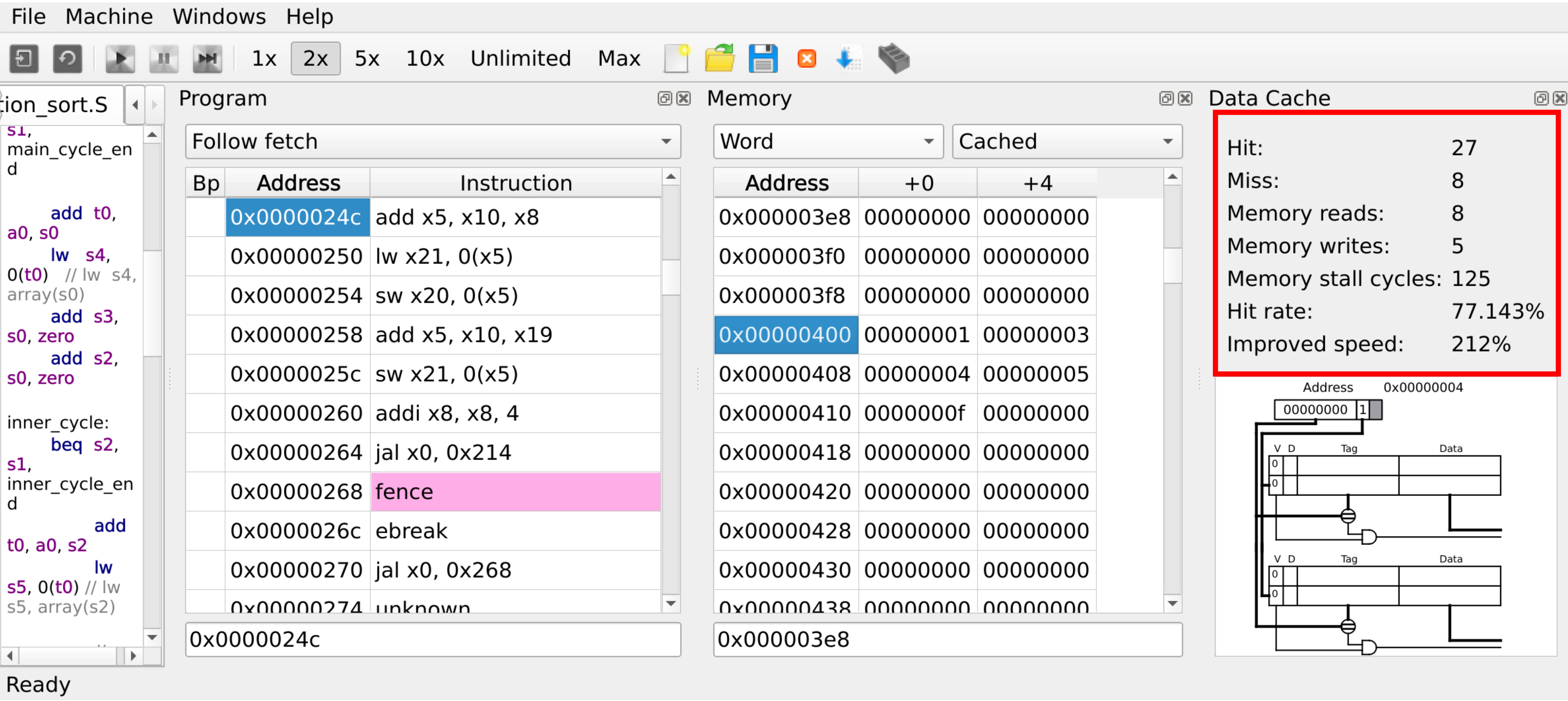
| Address | +0 | +4 |
|---|---|---|
| 0x000003e8 | 00000000 | 00000000 |
| 0x000003f0 | 00000000 | 00000000 |
| 0x000003f8 | 00000000 | 00000000 |
| 0x00000400 | 00000001 | 00000003 |
| 0x00000408 | 00000004 | 00000005 |
| 0x00000410 | 0000000f | 00000000 |
| 0x00000418 | 00000000 | 00000000 |
| 0x00000420 | 00000000 | 00000000 |
| 0x00000428 | 00000000 | 00000000 |
| 0x00000430 | 00000000 | 00000000 |
| 0x00000438 | 00000000 | 00000000 |

0x000003e8

Data Cache

| | |
|---|---|
| Hit: | 27 |
| Miss: | 8 |
| Memory reads: | 8 |
| Memory writes: | 5 |
| Memory stall cycles: | 125 |
| Hit rate: | 77.143% |
| Improved speed: | 212% |

# Example: Selection Sort 11 - Fence



```
s1,
main_cycle_en
d

        add  t0,
a0, s0
        lw   s4,
0(t0)   // lw  s4,
array(s0)
        add  s3,
s0, zero
        add  s2,
s0, zero

inner_cycle:
        beq  s2,
s1,
inner_cycle_en
d
             add
t0, a0, s2
             lw
s5, 0(t0) // lw
s5, array(s2)
```

| Bp | Address | Instruction |
|---|---|---|
| | 0x0000024c | add x5, x10, x8 |
| | 0x00000250 | lw x21, 0(x5) |
| | 0x00000254 | sw x20, 0(x5) |
| | 0x00000258 | add x5, x10, x19 |
| | 0x0000025c | sw x21, 0(x5) |
| | 0x00000260 | addi x8, x8, 4 |
| | 0x00000264 | jal x0, 0x214 |
| | 0x00000268 | fence |
| | 0x0000026c | ebreak |
| | 0x00000270 | jal x0, 0x268 |
| | 0x00000274 | unknown |

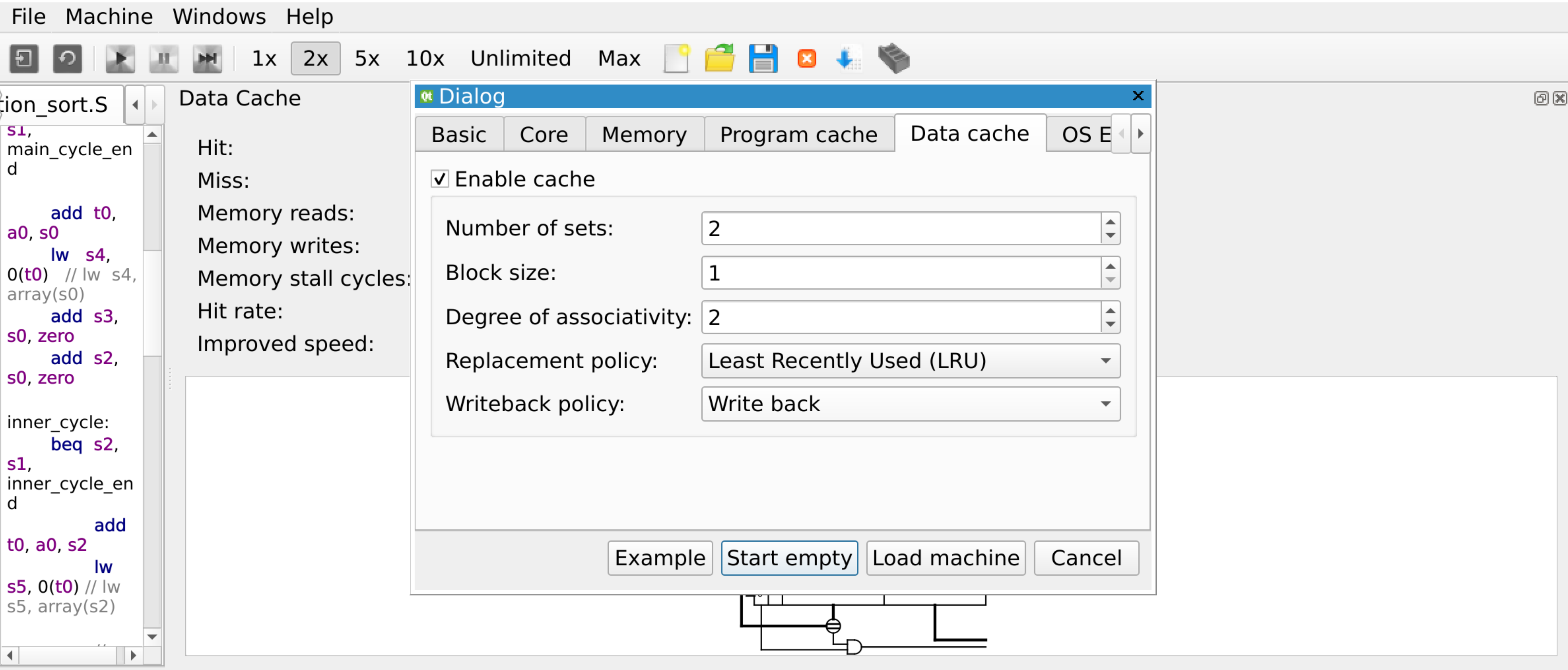
0x0000024c

| Address | +0 | +4 |
|---|---|---|
| 0x000003e8 | 00000000 | 00000000 |
| 0x000003f0 | 00000000 | 00000000 |
| 0x000003f8 | 00000000 | 00000000 |
| 0x00000400 | 00000001 | 00000003 |
| 0x00000408 | 00000004 | 00000005 |
| 0x00000410 | 0000000f | 00000000 |
| 0x00000418 | 00000000 | 00000000 |
| 0x00000420 | 00000000 | 00000000 |
| 0x00000428 | 00000000 | 00000000 |
| 0x00000430 | 00000000 | 00000000 |
| 0x00000438 | 00000000 | 00000000 |

0x000003e8

| | |
|---|---|
| Hit: | 27 |
| Miss: | 8 |
| Memory reads: | 8 |
| Memory writes: | 5 |
| Memory stall cycles: | 125 |
| Hit rate: | 77.143% |
| Improved speed: | 212% |

# Cache Configuration

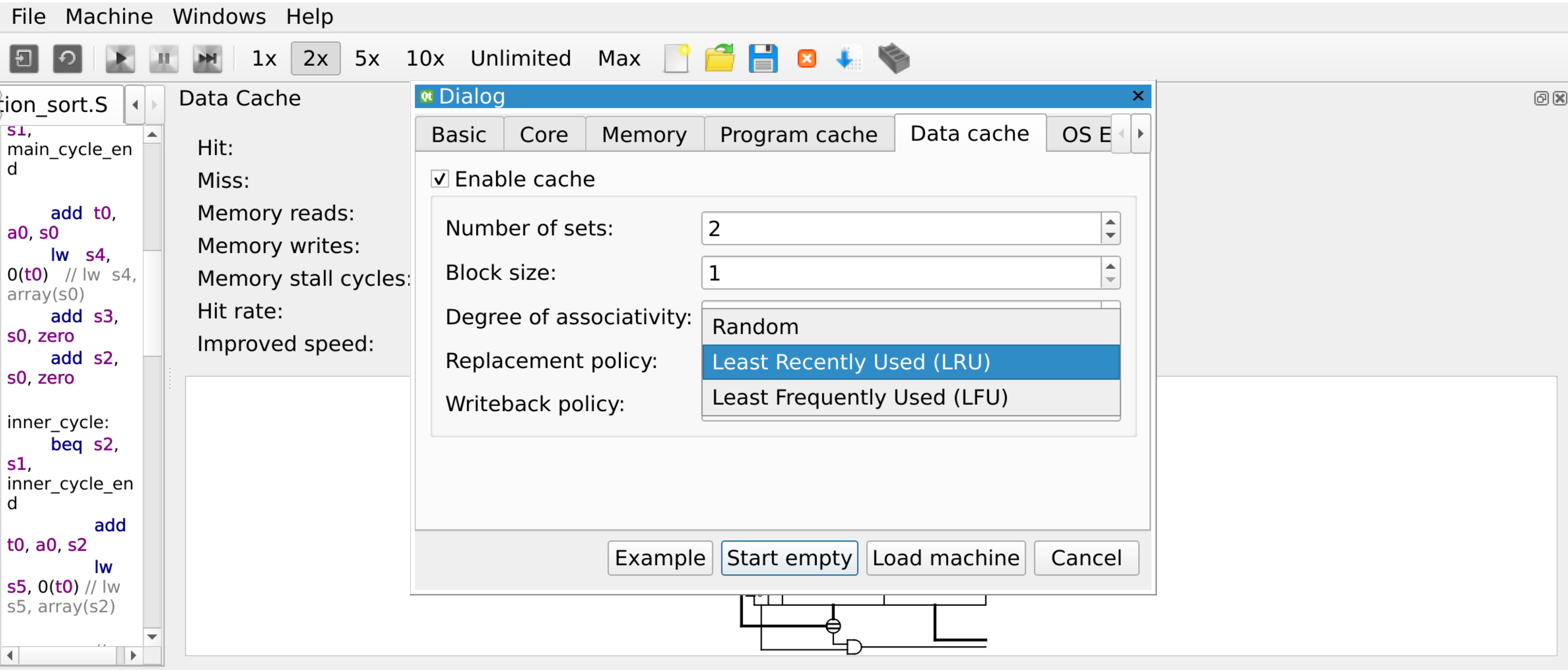


```
s1,
main_cycle_en
d

        add  t0,
a0, s0
        lw   s4,
0(t0)   // lw  s4,
array(s0)
        add  s3,
s0, zero
        add  s2,
s0, zero

inner_cycle:
        beq  s2,
s1,
inner_cycle_en
d
                add
t0, a0, s2
                lw
s5, 0(t0) // lw
s5, array(s2)
```

[✓] Enable cache

| | |
|---|---|
| Number of sets: | 2 |
| Block size: | 1 |
| Degree of associativity: | 2 |
| Replacement policy: | Least Recently Used (LRU) |
| Writeback policy: | Write back |

# Cache Configuration: Replacement Policy


File Machine Windows Help
1x 2x 5x 10x Unlimited Max
tion_sort.S
Data Cache
Dialog
Basic Core Memory Program cache Data cache OS E
Example Start empty Load machine Cancel

```
s1,
main_cycle_end

    add  t0, a0, s0
    lw   s4, 0(t0)   // lw  s4, array(s0)
    add  s3, s0, zero
    add  s2, s0, zero

inner_cycle:
    beq  s2, s1, inner_cycle_end
        add t0, a0, s2
        lw  s5, 0(t0) // lw s5, array(s2)
```

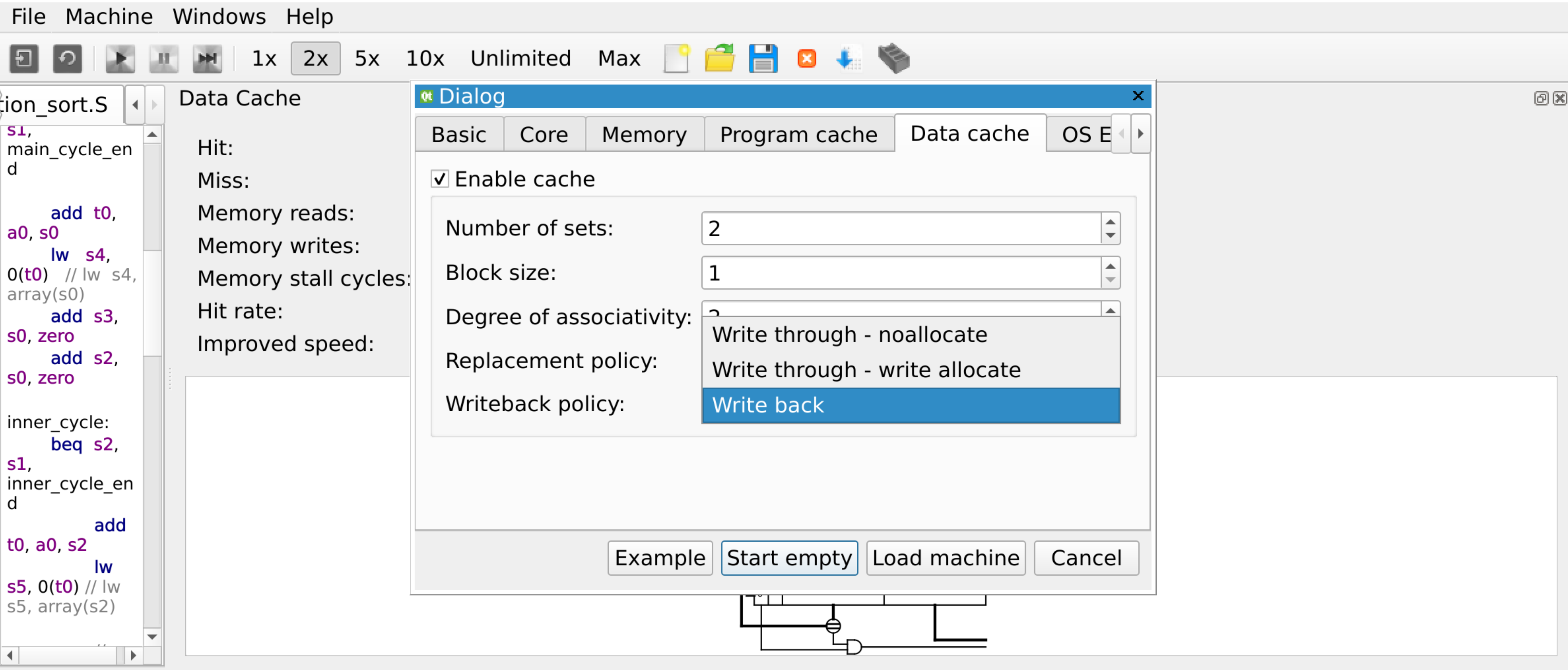
Hit:
Miss:
Memory reads:
Memory writes:
Memory stall cycles:
Hit rate:
Improved speed:

✓ Enable cache

Number of sets: 2
Block size: 1
Degree of associativity:
Replacement policy:
Writeback policy:

Random
Least Recently Used (LRU)
Least Frequently Used (LFU)

# Cache Configuration: Writeback Policy



```
s1,
main_cycle_en
d

        add  t0,
a0, s0
        lw   s4,
0(t0)   // lw  s4,
array(s0)
        add  s3,
s0, zero
        add  s2,
s0, zero

inner_cycle:
        beq  s2,
s1,
inner_cycle_en
d
            add
t0, a0, s2
            lw
s5, 0(t0) // lw
s5, array(s2)
```

☑ Enable cache

Number of sets: 2

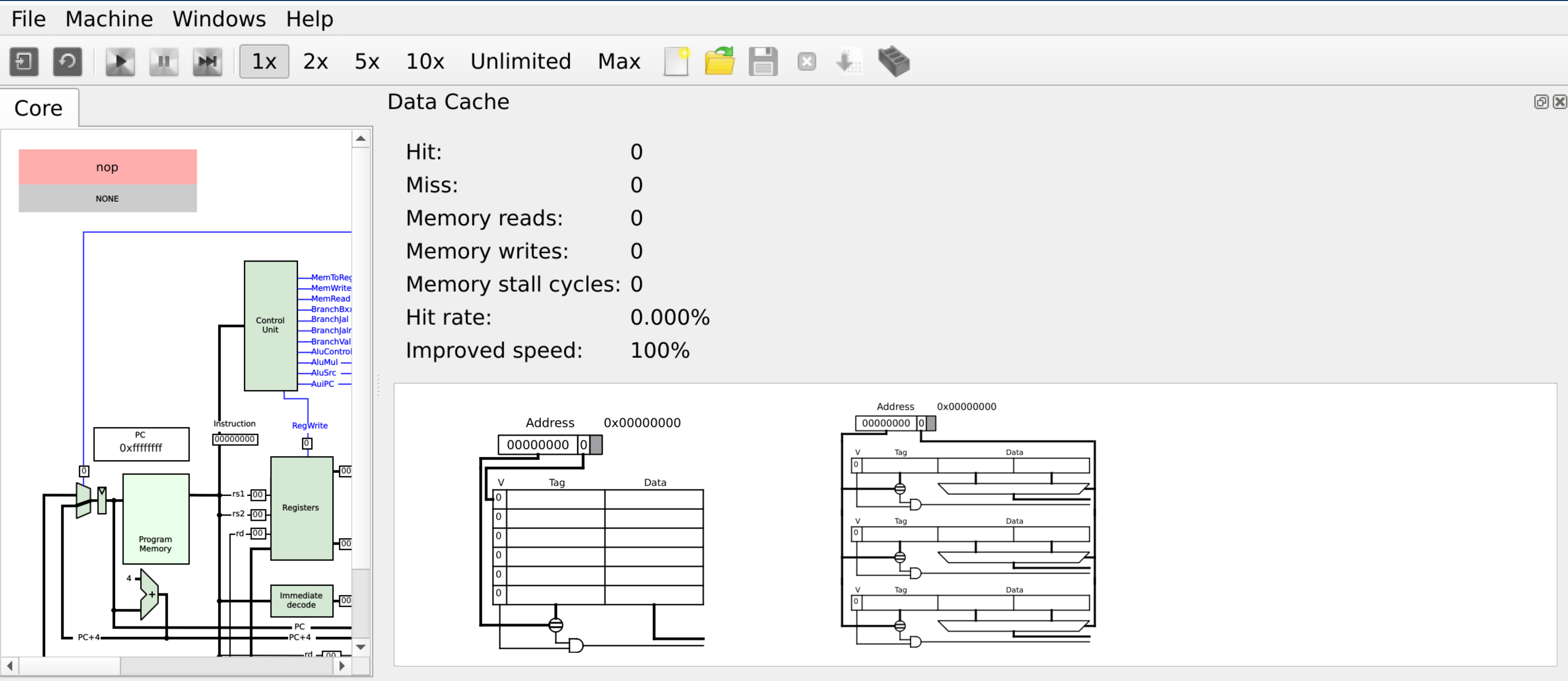
Block size: 1

Degree of associativity: 2

Replacement policy:

Writeback policy:
- Write through - noallocate
- Write through - write allocate
- [✓ Write back]

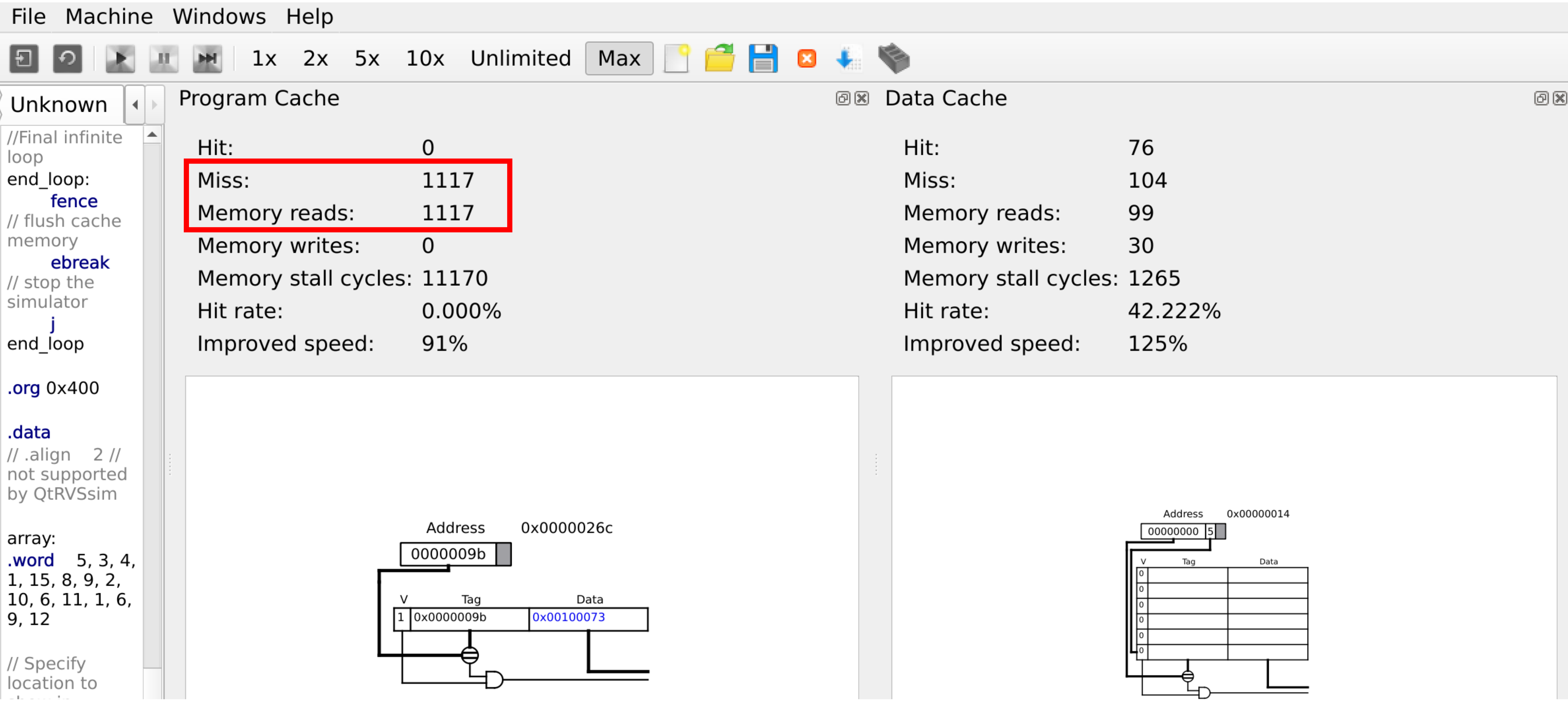
# Cache Shape Visualization



Data Cache

Hit: 0
Miss: 0
Memory reads: 0
Memory writes: 0
Memory stall cycles: 0
Hit rate: 0.000%
Improved speed: 100%

# Program Cache



File Machine Windows Help

1x 2x 5x 10x Unlimited Max

```
Unknown
//Final infinite
loop
end_loop:
        fence
// flush cache
memory
        ebreak
// stop the
simulator
        j
end_loop

.org 0x400

.data
// .align    2 //
not supported
by QtRVSsim

array:
.word    5, 3, 4,
1, 15, 8, 9, 2,
10, 6, 11, 1, 6,
9, 12

// Specify
location to
```

Program Cache

```
Hit:                  0
Miss:                 1117
Memory reads:         1117
Memory writes:        0
Memory stall cycles:  11170
Hit rate:             0.000%
Improved speed:       91%
```

```
Address      0x0000026c
0000009b
V  Tag          Data
1  0x0000009b   0x00100073
```

Data Cache

```
Hit:                  76
Miss:                 104
Memory reads:         99
Memory writes:        30
Memory stall cycles:  1265
Hit rate:             42.222%
Improved speed:       125%
```

```
Address      0x00000014
00000000 5
V  Tag  Data
0
0
0
0
0
0
```

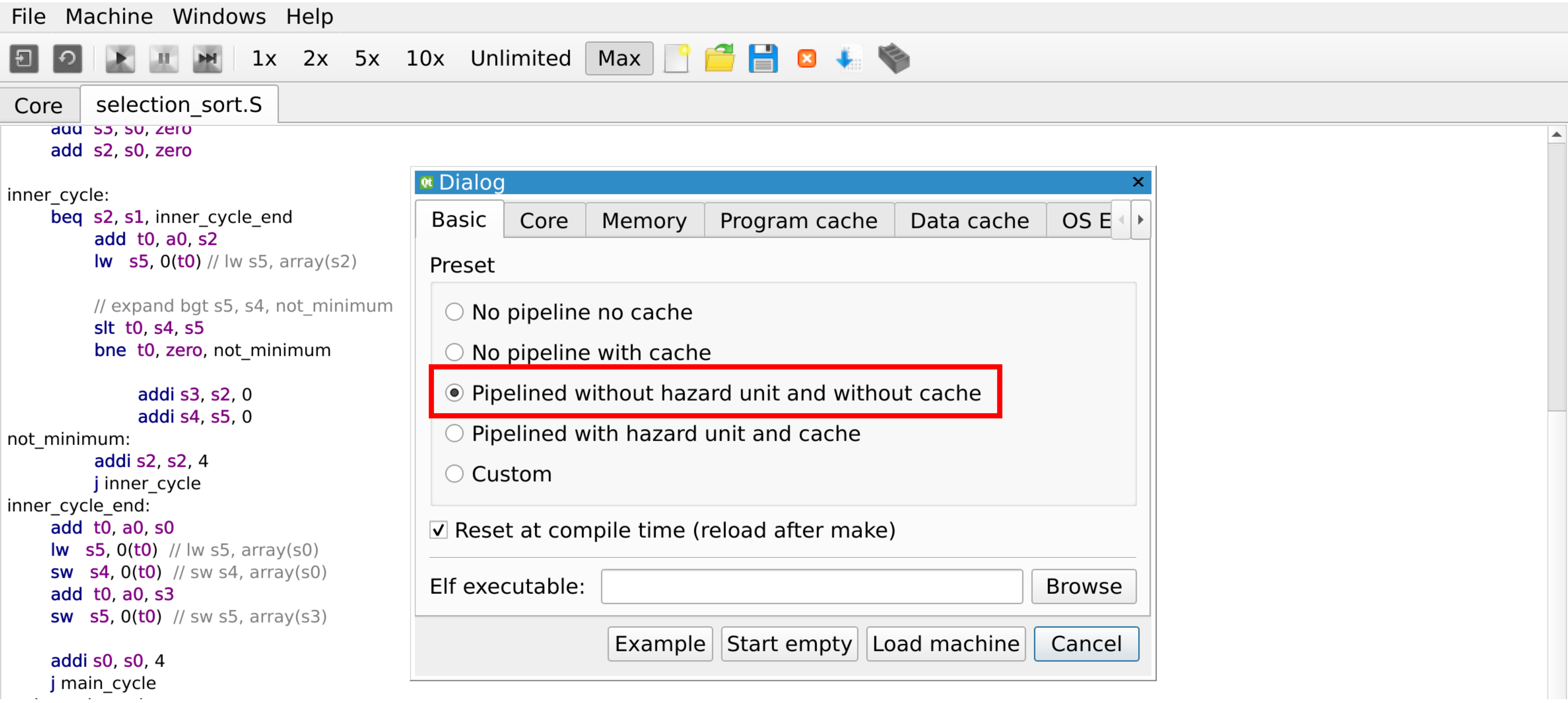
## Simulator Pragmas

```
#pragma qtrvsim show registers
#pragma qtrvsim show memory
#pragma qtrvsim focus memory array
```

# Pipeline

Interstage Registers
Data Hazards
Forwarding

# Enabling Pipeline


File Machine Windows Help
1x 2x 5x 10x Unlimited Max
Core selection_sort.S

```
    add  s3, s0, zero
    add  s2, s0, zero

inner_cycle:
    beq  s2, s1, inner_cycle_end
        add  t0, a0, s2
        lw   s5, 0(t0) // lw s5, array(s2)

        // expand bgt s5, s4, not_minimum
        slt  t0, s4, s5
        bne  t0, zero, not_minimum

            addi s3, s2, 0
            addi s4, s5, 0
not_minimum:
        addi s2, s2, 4
        j inner_cycle
inner_cycle_end:
    add  t0, a0, s0
    lw   s5, 0(t0)  // lw s5, array(s0)
    sw   s4, 0(t0)  // sw s4, array(s0)
    add  t0, a0, s3
    sw   s5, 0(t0)  // sw s5, array(s3)

    addi s0, s0, 4
    j main_cycle
```

Dialog

Basic | Core | Memory | Program cache | Data cache | OS E

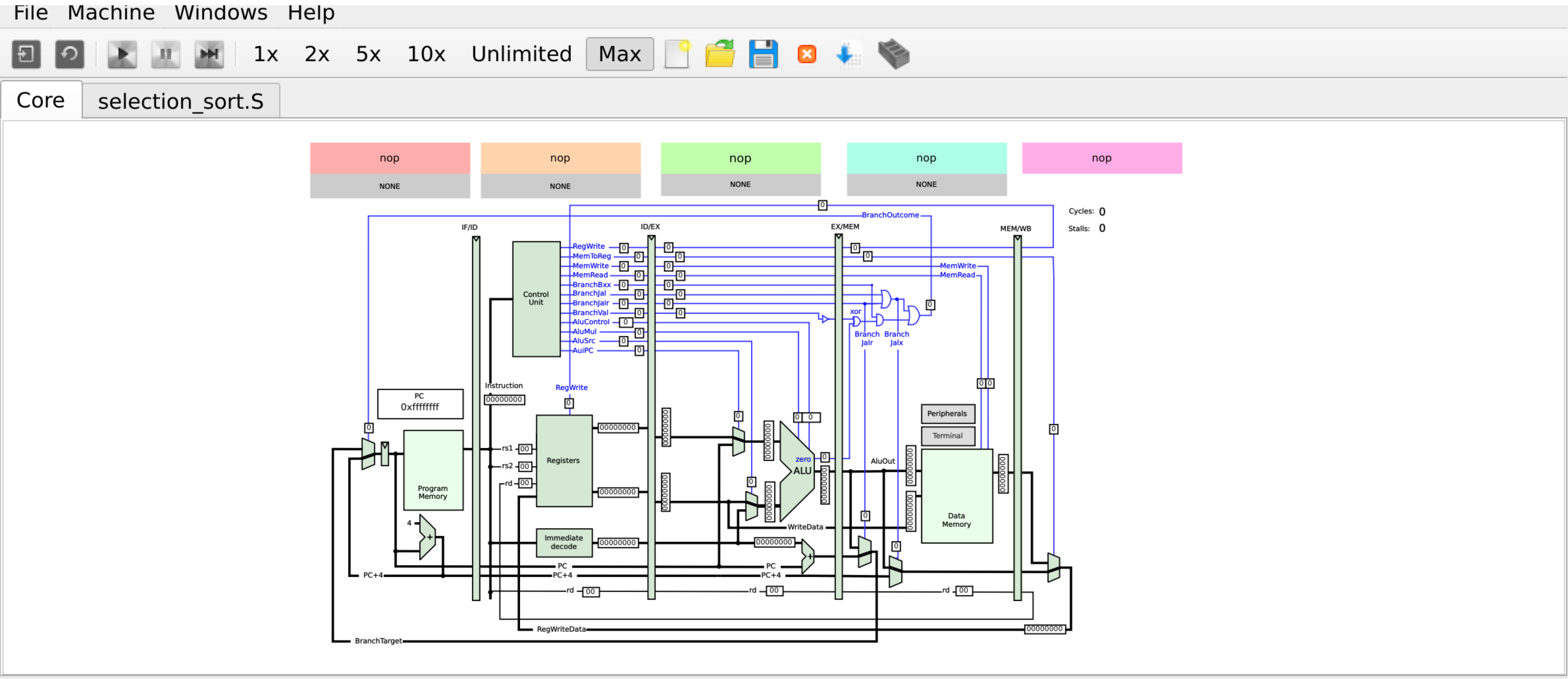
Preset

- ( ) No pipeline no cache
- ( ) No pipeline with cache
- (●) Pipelined without hazard unit and without cache
- ( ) Pipelined with hazard unit and cache
- ( ) Custom

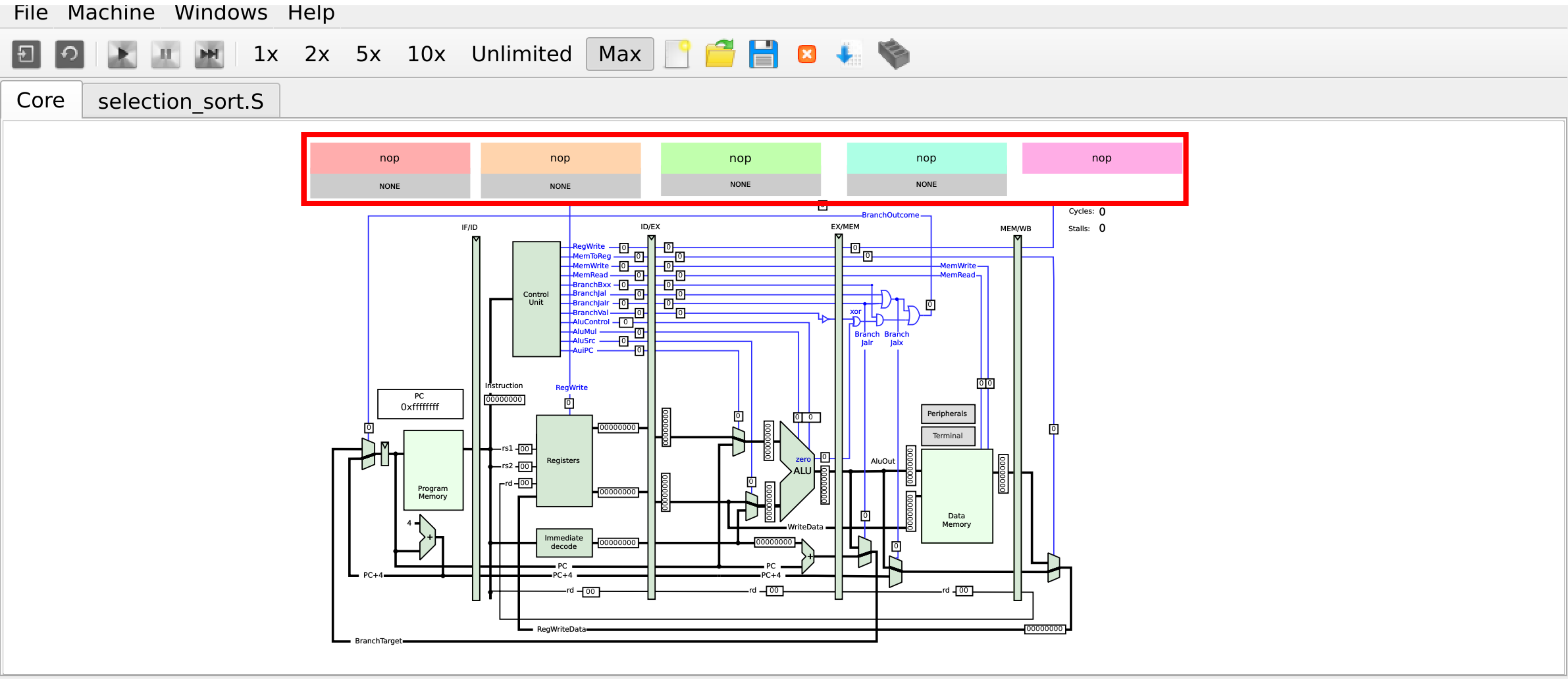
[✓] Reset at compile time (reload after make)

Elf executable: Browse

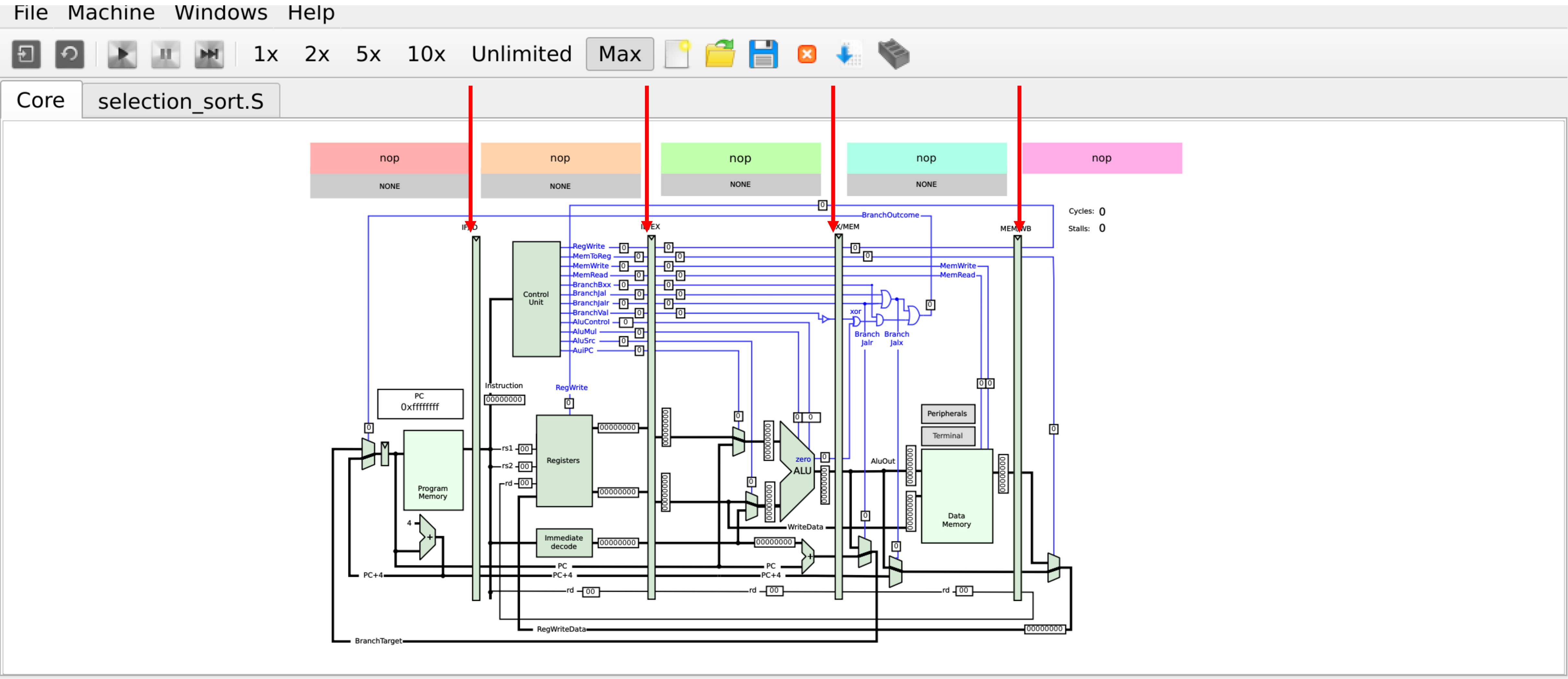
# Pipeline


File
Machine
Windows
Help
1x
2x
5x
10x
Unlimited
Max
Core
selection_sort.S
nop
NONE
nop
NONE
nop
NONE
nop
NONE
nop
IF/ID
ID/EX
EX/MEM
MEM/WB
BranchOutcome
Cycles: 0
Stalls: 0
RegWrite
MemToReg
MemWrite
MemRead
BranchBxx
BranchJal
BranchJalr
BranchVal
AluControl
AluMul
AluSrc
AuiPC
Control Unit
MemWrite
MemRead
xor
Branch Jalr
Branch Jalx
Instruction
00000000
RegWrite
PC
0xffffffff
Program Memory
Registers
rs1
rs2
rd
Immediate decode
zero
ALU
AluOut
WriteData
Peripherals
Terminal
Data Memory
PC
PC+4
rd
RegWriteData
BranchTarget

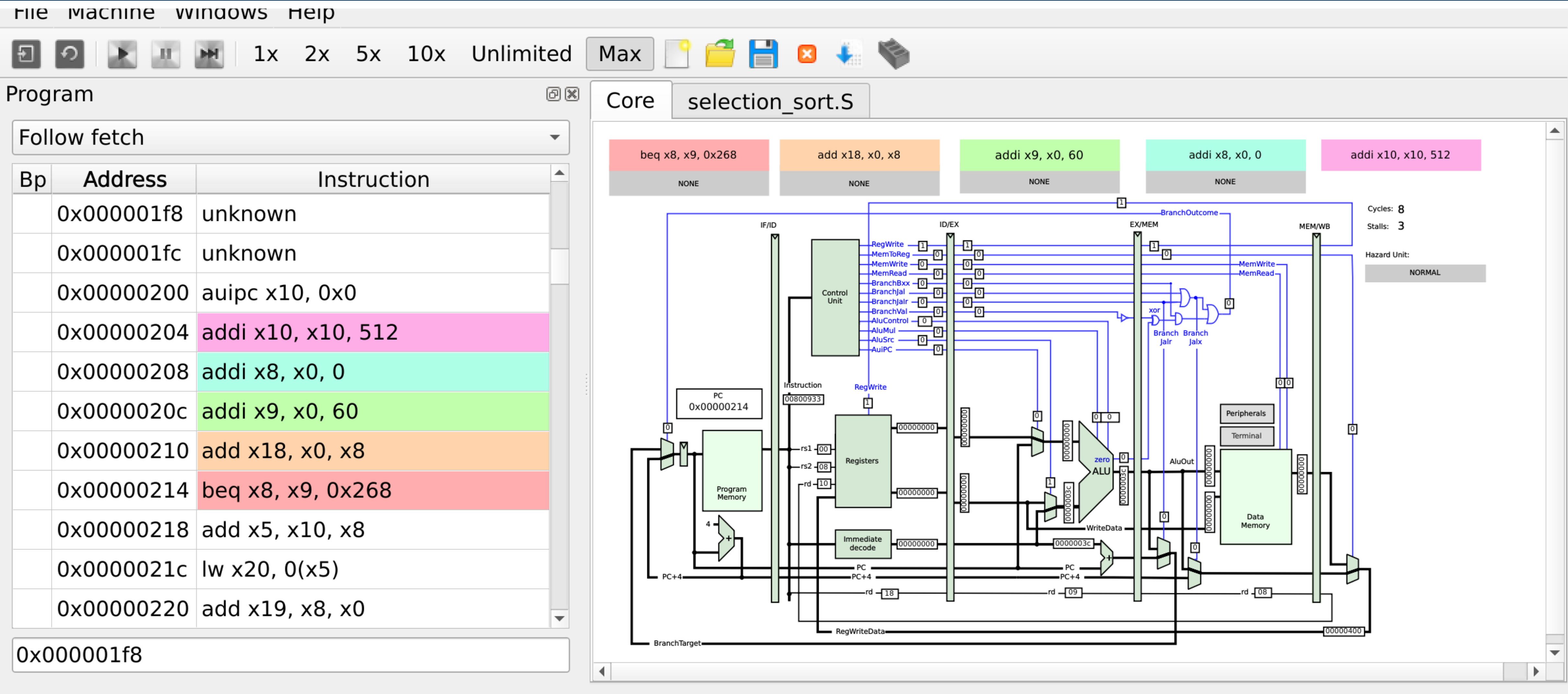
# Pipeline


File
Machine
Windows
Help
1x
2x
5x
10x
Unlimited
Max
Core
selection_sort.S
nop
NONE
nop
NONE
nop
NONE
nop
NONE
nop
Cycles: 0
Stalls: 0
IF/ID
ID/EX
EX/MEM
MEM/WB
BranchOutcome
Control Unit
RegWrite
MemToReg
MemWrite
MemRead
BranchBxx
BranchJal
BranchJalr
BranchVal
AluControl
AluMul
AluSrc
AuiPC
xor
Branch Jalr
Branch Jalx
PC
0xffffffff
Instruction
00000000
RegWrite
Program Memory
Registers
rs1
rs2
rd
Immediate decode
zero
ALU
WriteData
AluOut
Peripherals
Terminal
Data Memory
PC+4
BranchTarget
RegWriteData

# Pipeline


File
Machine
Windows
Help
1x
2x
5x
10x
Unlimited
Max
Core
selection_sort.S
nop
NONE
nop
NONE
nop
NONE
nop
NONE
nop
IF/ID
ID/EX
EX/MEM
MEM/WB
Cycles: 0
Stalls: 0
Control Unit
RegWrite
MemToReg
MemWrite
MemRead
BranchBxx
BranchJal
BranchJalr
BranchVal
AluControl
AluMul
AluSrc
AuiPC
BranchOutcome
MemWrite
MemRead
xor
Branch Jalr
Branch Jalx
PC
0xffffffff
Instruction
00000000
RegWrite
Program Memory
Registers
rs1
rs2
rd
Immediate decode
zero
ALU
WriteData
AluOut
Peripherals
Terminal
Data Memory
PC
PC+4
rd
RegWriteData
BranchTarget

# Program View



File Machine Windows Help

1x 2x 5x 10x Unlimited Max

Program

Follow fetch

| Bp | Address | Instruction |
|---|---|---|
| | 0x000001f8 | unknown |
| | 0x000001fc | unknown |
| | 0x00000200 | auipc x10, 0x0 |
| | 0x00000204 | addi x10, x10, 512 |
| | 0x00000208 | addi x8, x0, 0 |
| | 0x0000020c | addi x9, x0, 60 |
| | 0x00000210 | add x18, x0, x8 |
| | 0x00000214 | beq x8, x9, 0x268 |
| | 0x00000218 | add x5, x10, x8 |
| | 0x0000021c | lw x20, 0(x5) |
| | 0x00000220 | add x19, x8, x0 |

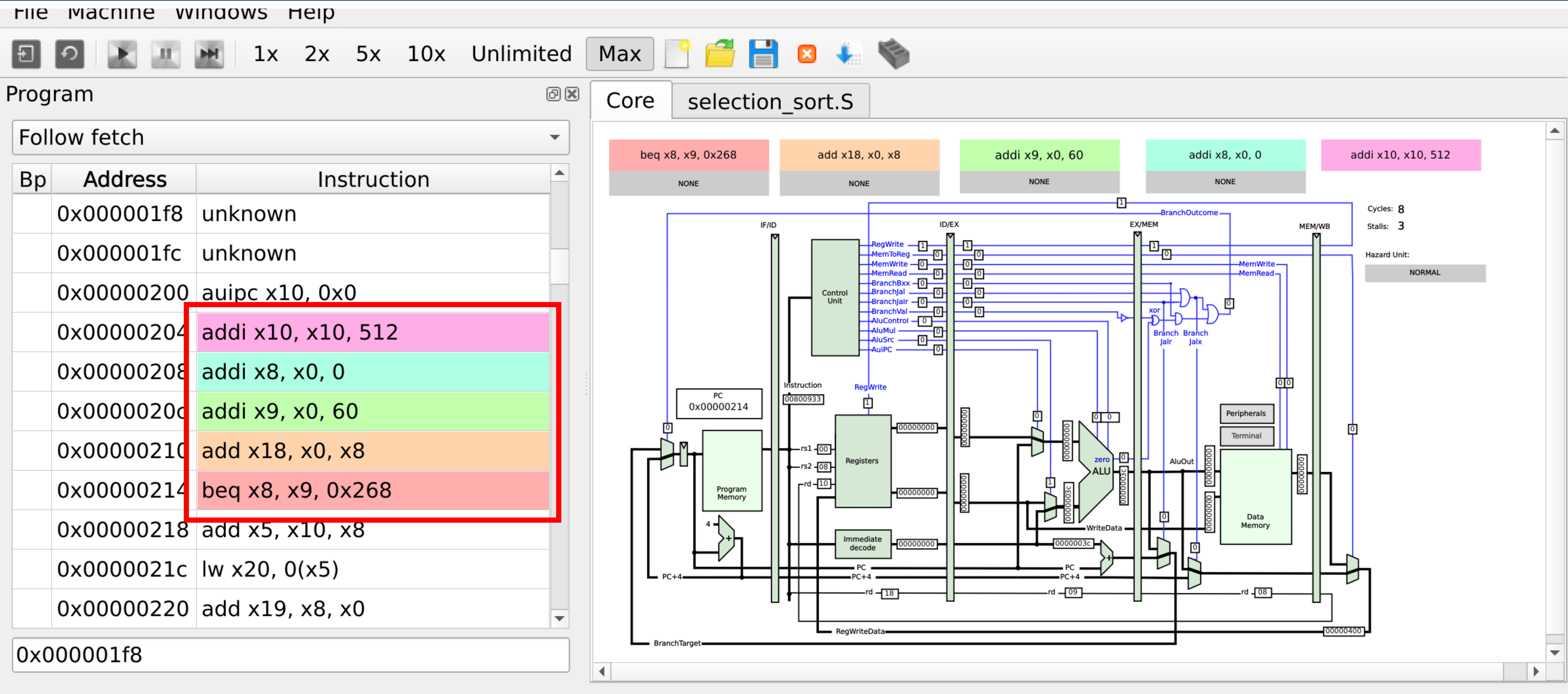
0x000001f8

Core | selection_sort.S

| beq x8, x9, 0x268 | add x18, x0, x8 | addi x9, x0, 60 | addi x8, x0, 0 | addi x10, x10, 512 |
|---|---|---|---|---|
| NONE | NONE | NONE | NONE | |

Cycles: 8
Stalls: 3
Hazard Unit: NORMAL

# Program View



Follow fetch

| Bp | Address | Instruction |
|---|---|---|
| | 0x000001f8 | unknown |
| | 0x000001fc | unknown |
| | 0x00000200 | auipc x10, 0x0 |
| | 0x00000204 | addi x10, x10, 512 |
| | 0x00000208 | addi x8, x0, 0 |
| | 0x0000020c | addi x9, x0, 60 |
| | 0x00000210 | add x18, x0, x8 |
| | 0x00000214 | beq x8, x9, 0x268 |
| | 0x00000218 | add x5, x10, x8 |
| | 0x0000021c | lw x20, 0(x5) |
| | 0x00000220 | add x19, x8, x0 |

0x000001f8

| beq x8, x9, 0x268 | add x18, x0, x8 | addi x9, x0, 60 | addi x8, x0, 0 | addi x10, x10, 512 |
|---|---|---|---|---|
| NONE | NONE | NONE | NONE | |

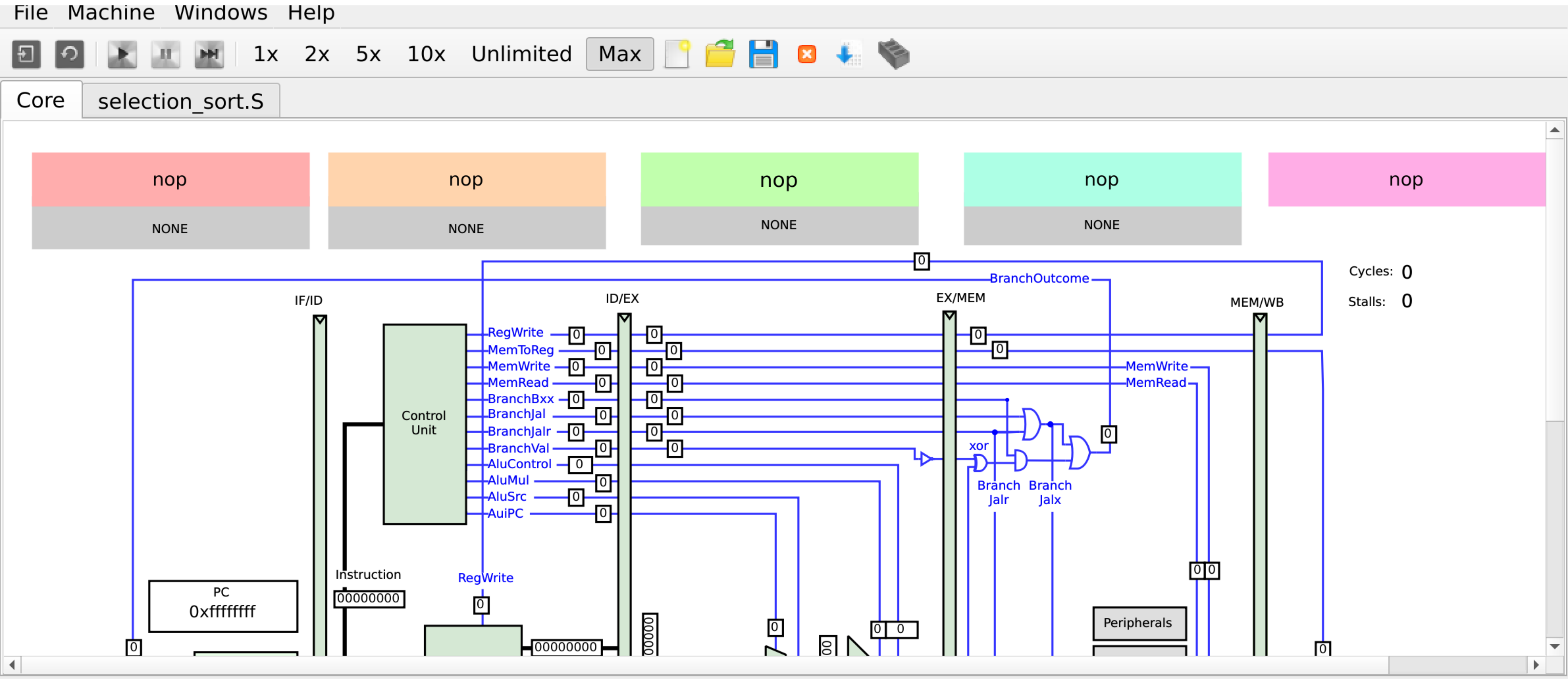
PC 0x00000214

Instruction 00800933

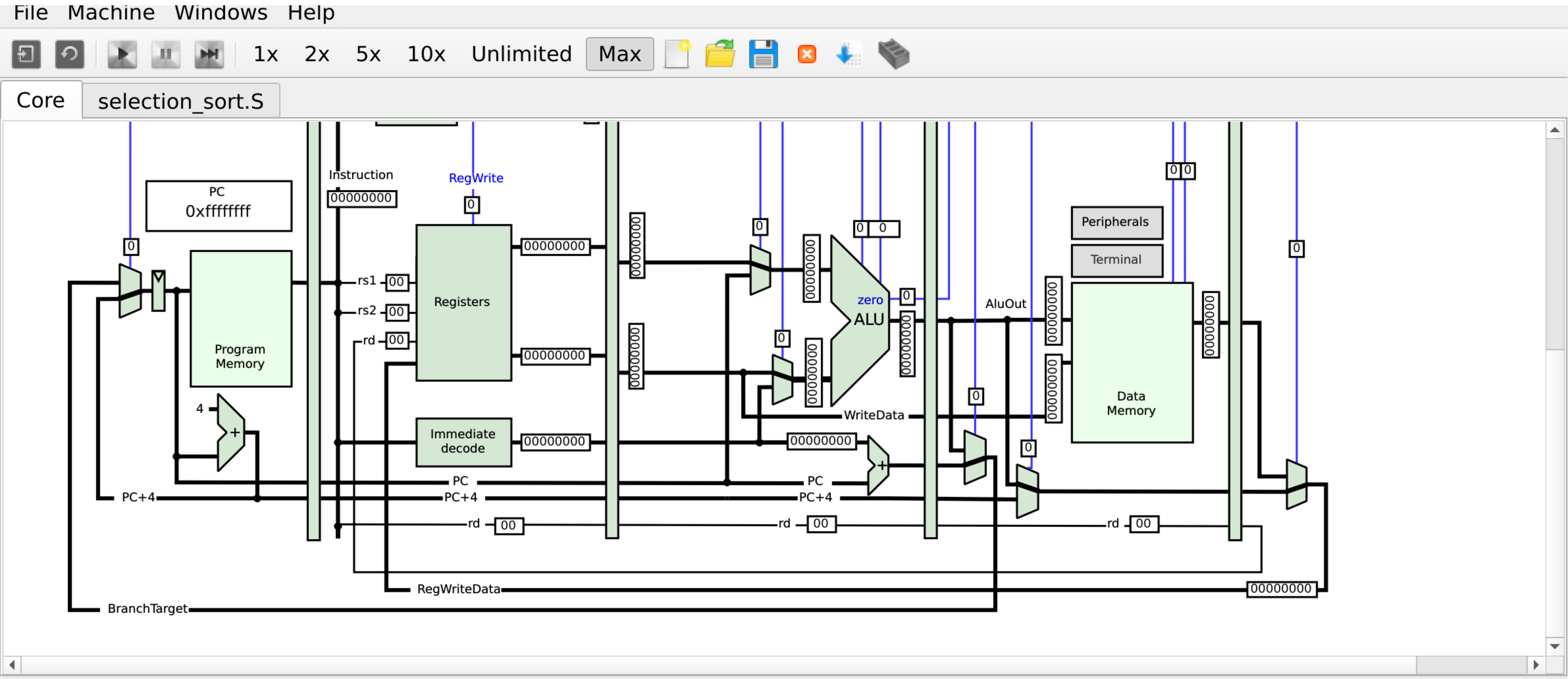
Cycles: 8
Stalls: 3

Hazard Unit: NORMAL

# Control Signals



# Data Signals



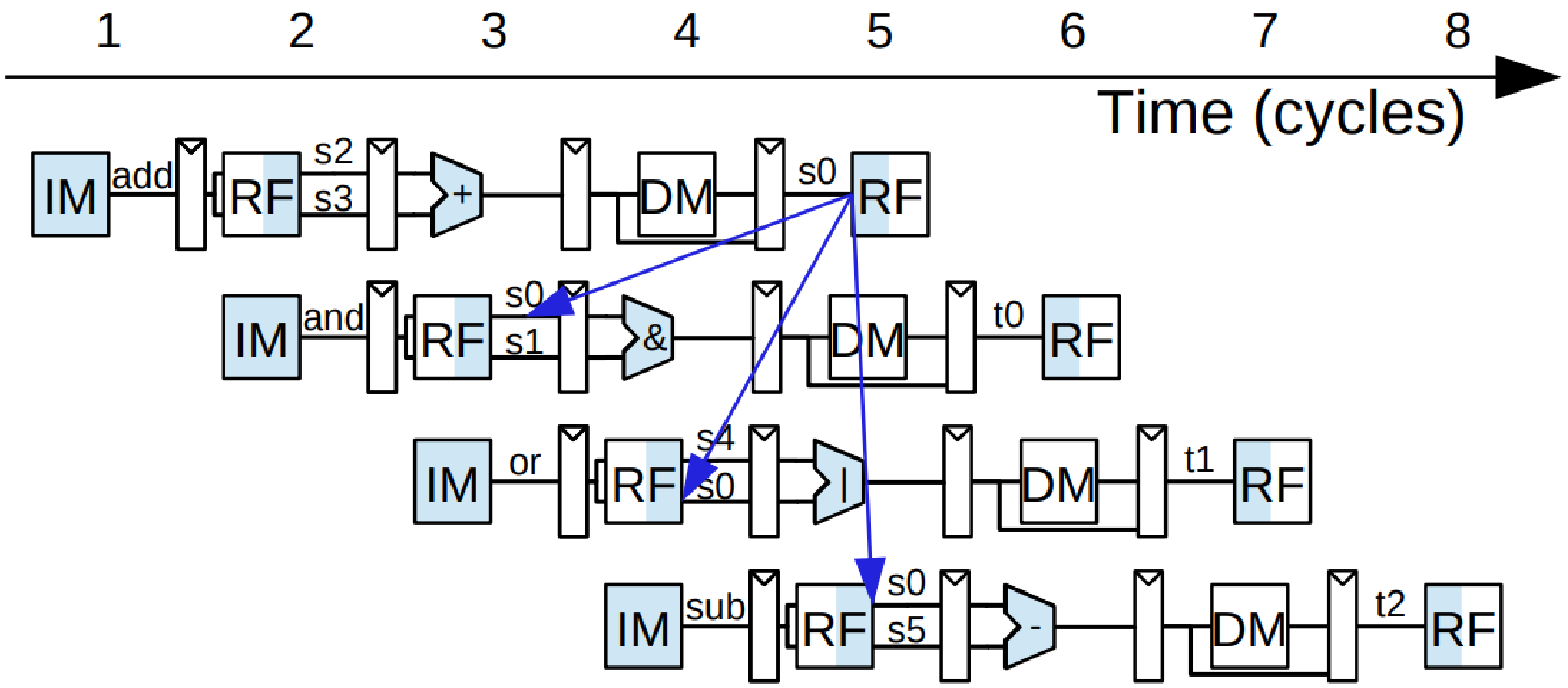
# Data Hazards

add **s0**, s2, s3

and t0, **s0**, s1

or t1, s4, **s0**

sub t2, **s0**, s5



https://cw.fel.cvut.cz/b222/_media/courses/b35apo/en/lectures/05/b35apo_lecture05-pipeline.pdf

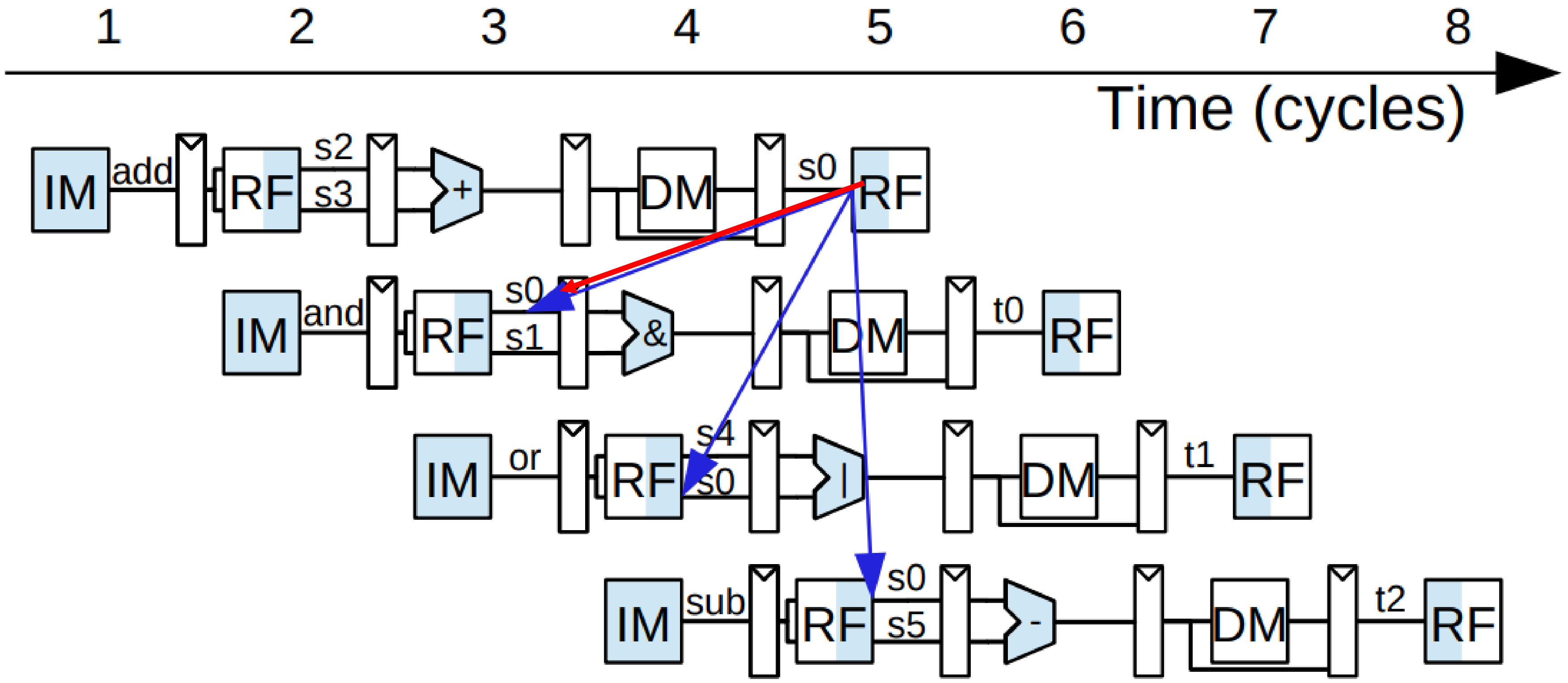
# Data Hazards

add **s0**, s2, s3

and t0, **s0**, s1

or t1, s4, **s0**

sub t2, **s0**, s5



https://cw.fel.cvut.cz/b222/_media/courses/b35apo/en/lectures/05/b35apo_lecture05-pipeline.pdf

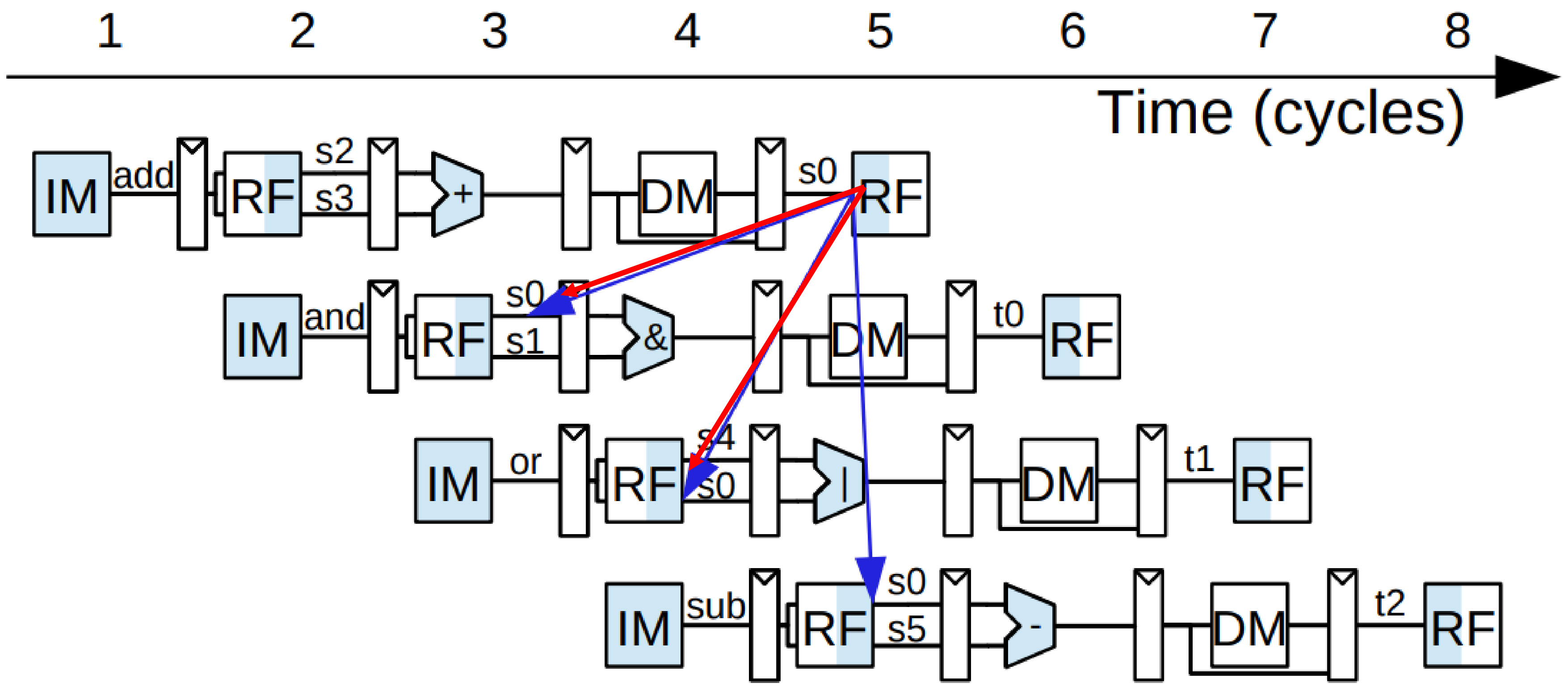
# Data Hazards

add **s0**, s2, s3

and t0, **s0**, s1

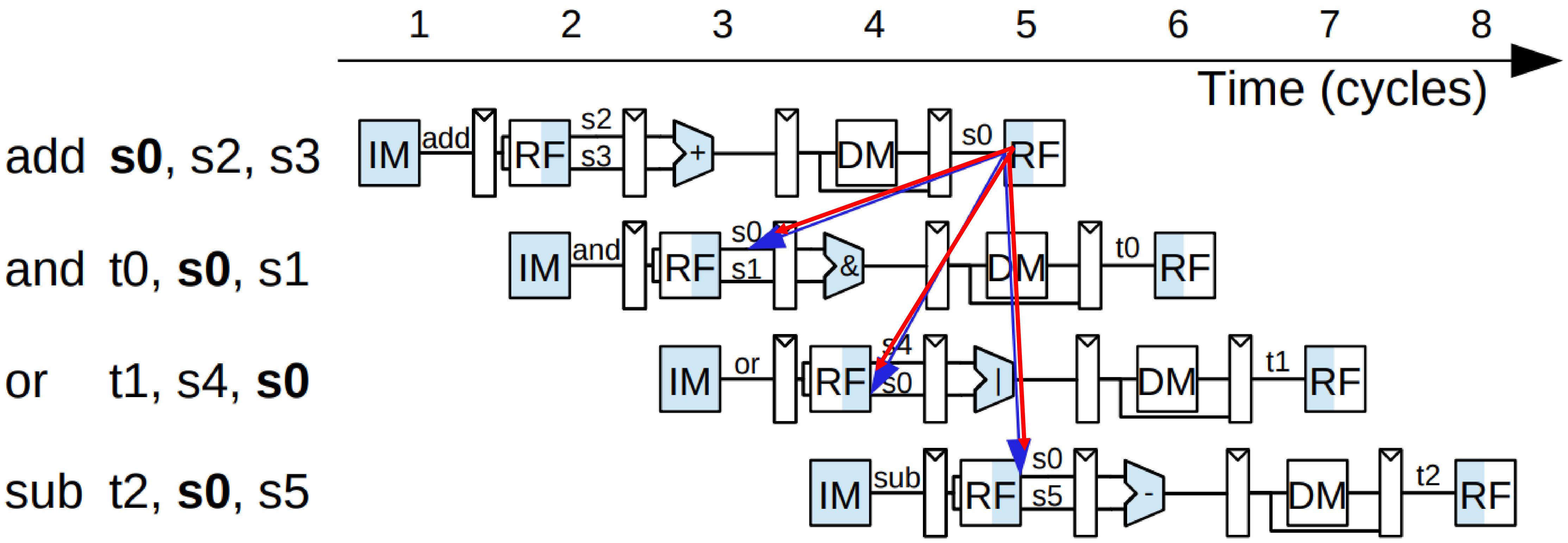
or t1, s4, **s0**

sub t2, **s0**, s5



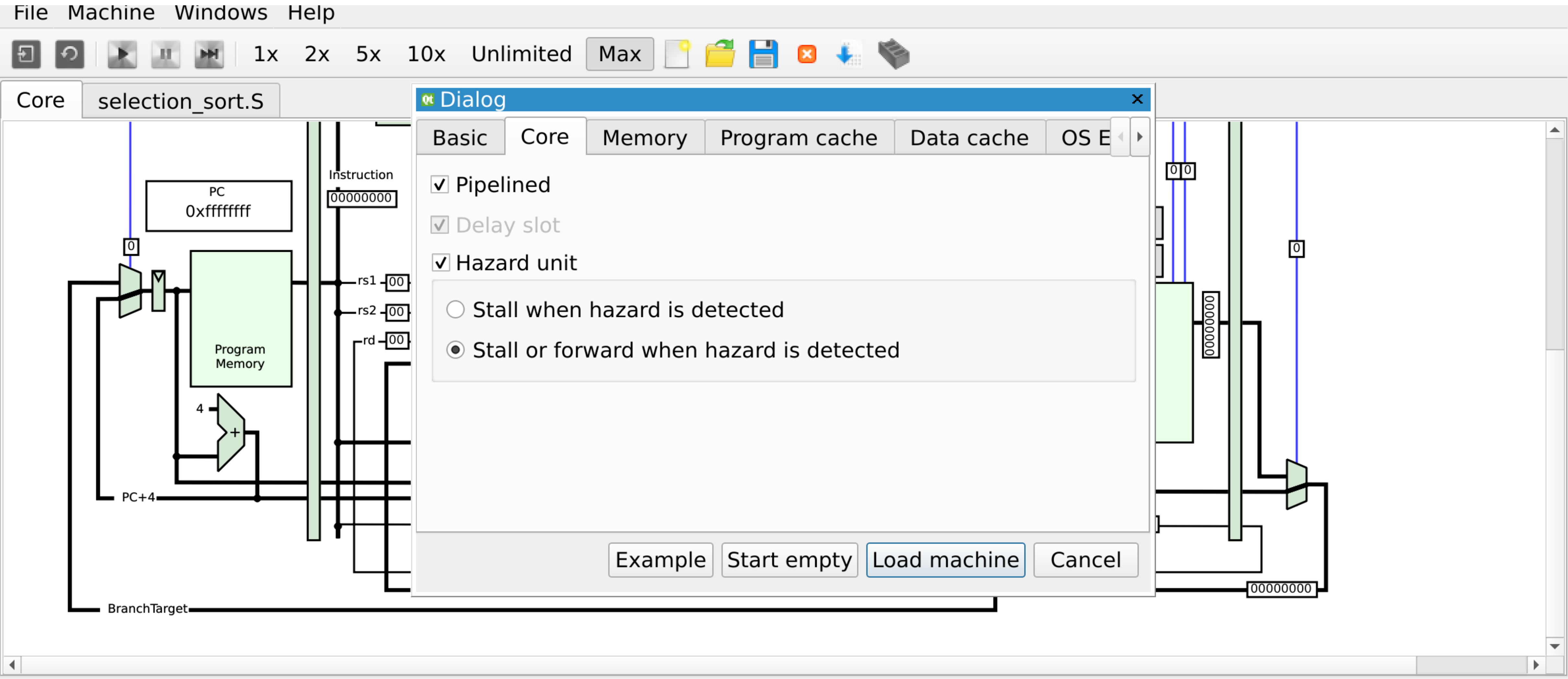
https://cw.fel.cvut.cz/b222/_media/courses/b35apo/en/lectures/05/b35apo_lecture05-pipeline.pdf

# Data Hazards



https://cw.fel.cvut.cz/b222/_media/courses/b35apo/en/lectures/05/b35apo_lecture05-pipeline.pdf

# Hazard Unit Configuration



[✓] Pipelined

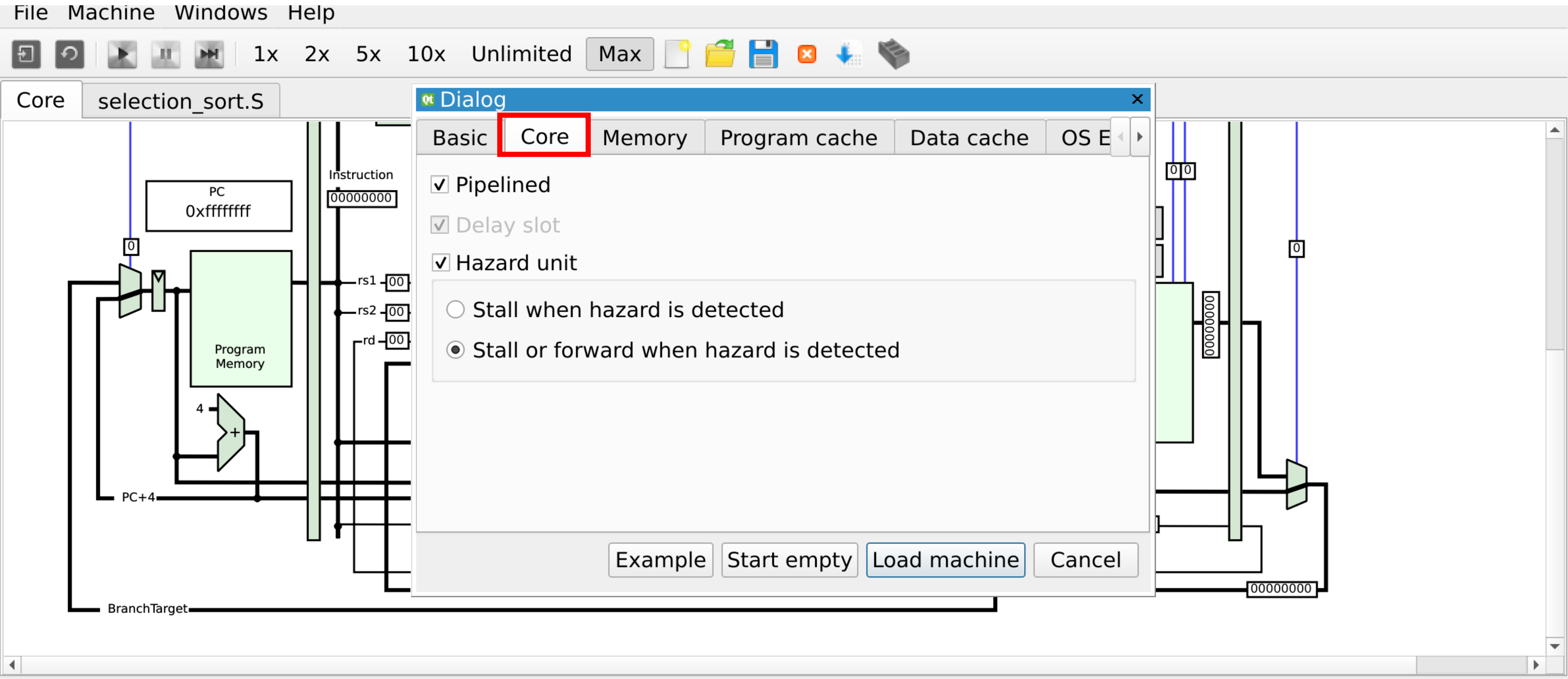
[✓] Delay slot

[✓] Hazard unit

○ Stall when hazard is detected

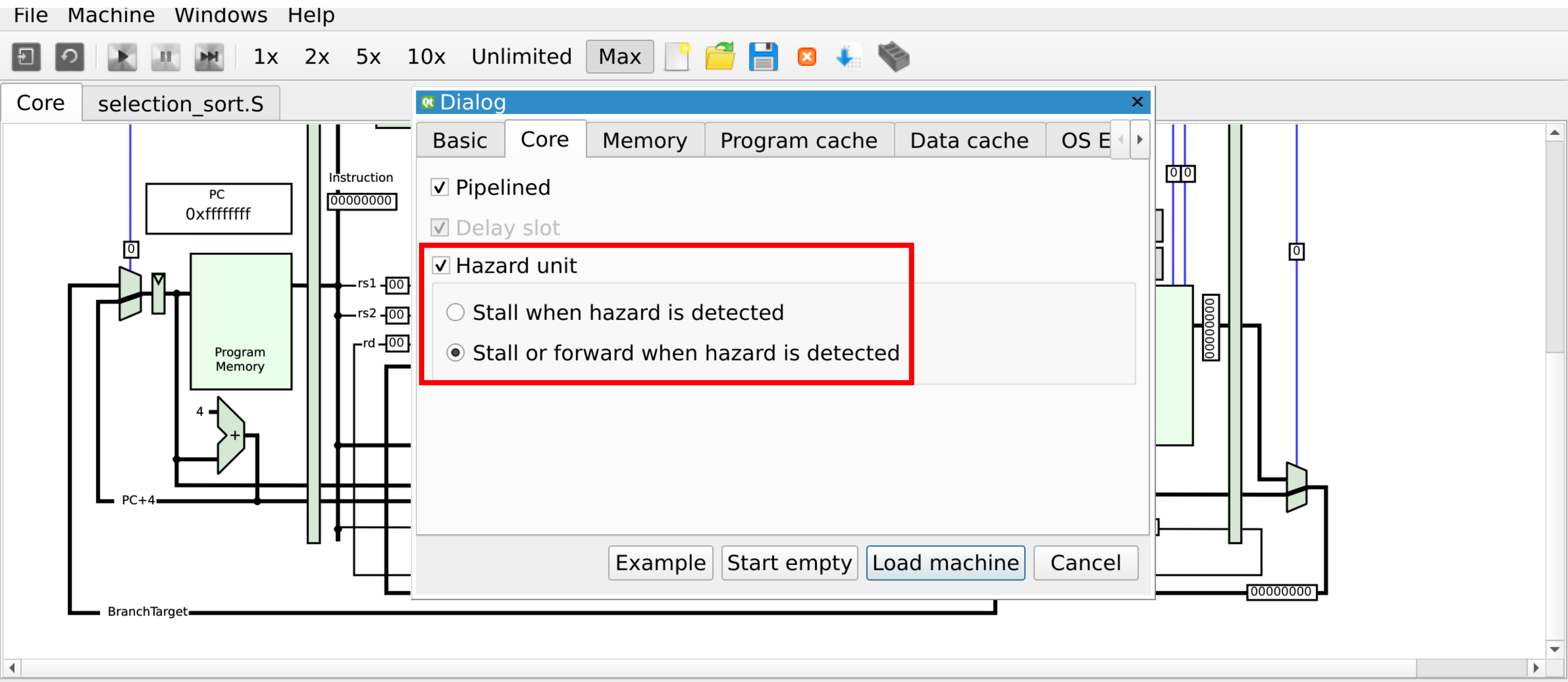
[✓ ● Stall or forward when hazard is detected]

## Hazard Unit Configuration


File Machine Windows Help
1x 2x 5x 10x Unlimited Max
Core selection_sort.S
PC
0xffffffff
Instruction
00000000
Program Memory
rs1
rs2
rd
PC+4
BranchTarget
Dialog
Basic Core Memory Program cache Data cache OS E
Example Start empty Load machine Cancel

[✓] Pipelined
[✓] Delay slot
[✓] Hazard unit
( ) Stall when hazard is detected
(●) Stall or forward when hazard is detected

# Hazard Unit Configuration


File Machine Windows Help
1x 2x 5x 10x Unlimited Max
Core
selection_sort.S
PC
0xffffffff
Instruction
00000000
Program Memory
rs1
rs2
rd
PC+4
BranchTarget
Dialog
Basic Core Memory Program cache Data cache OS E
Example Start empty Load machine Cancel

[✓] Pipelined

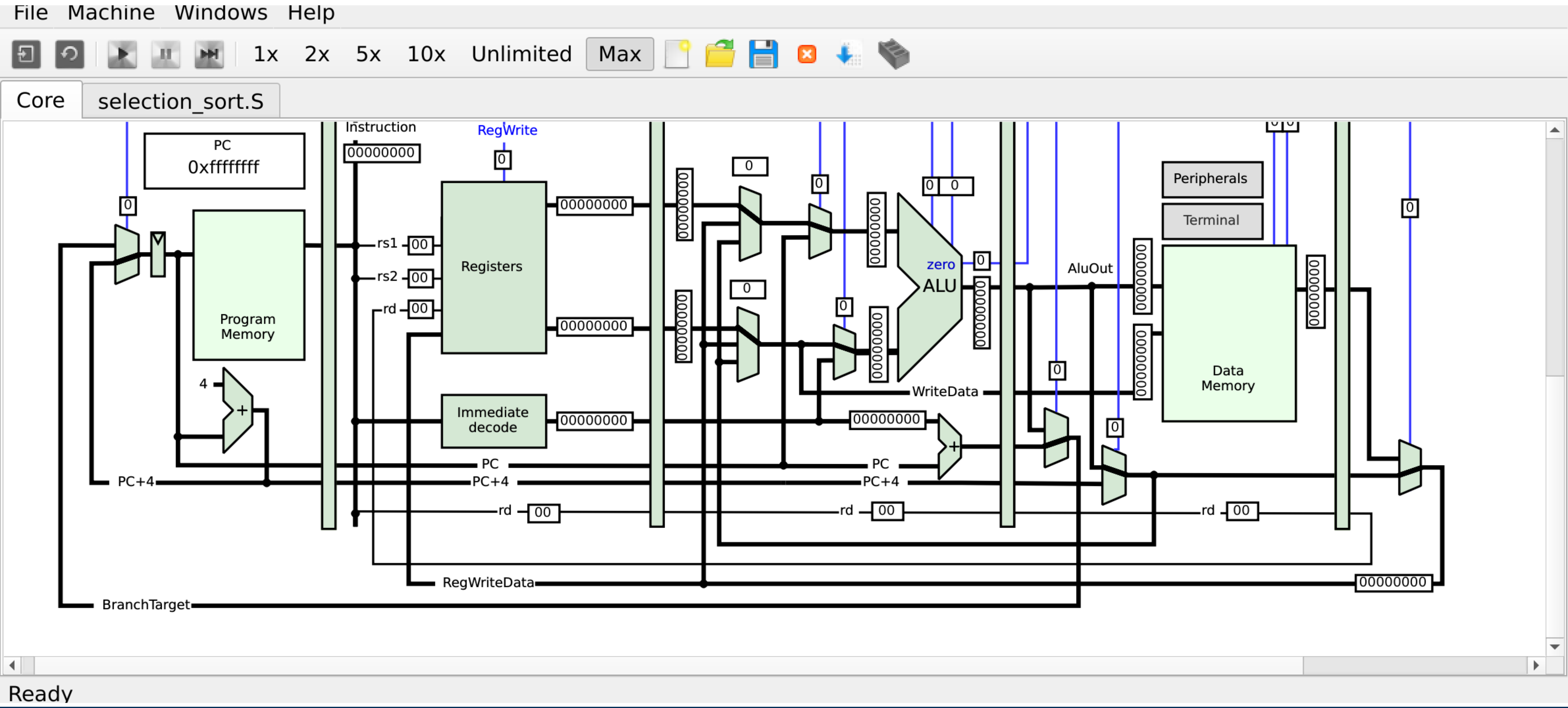
[✓] Delay slot

[✓] Hazard unit

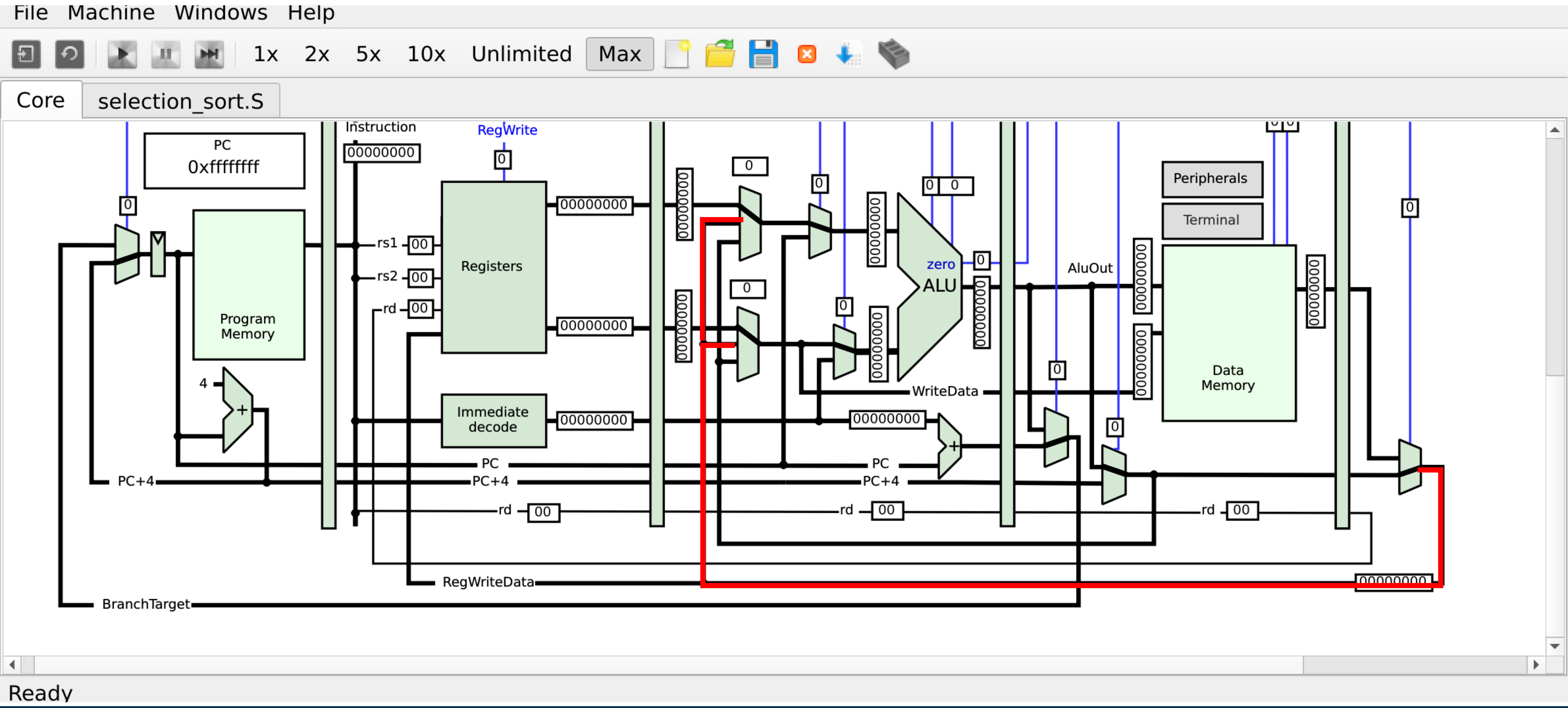
( ) Stall when hazard is detected

(●) Stall or forward when hazard is detected

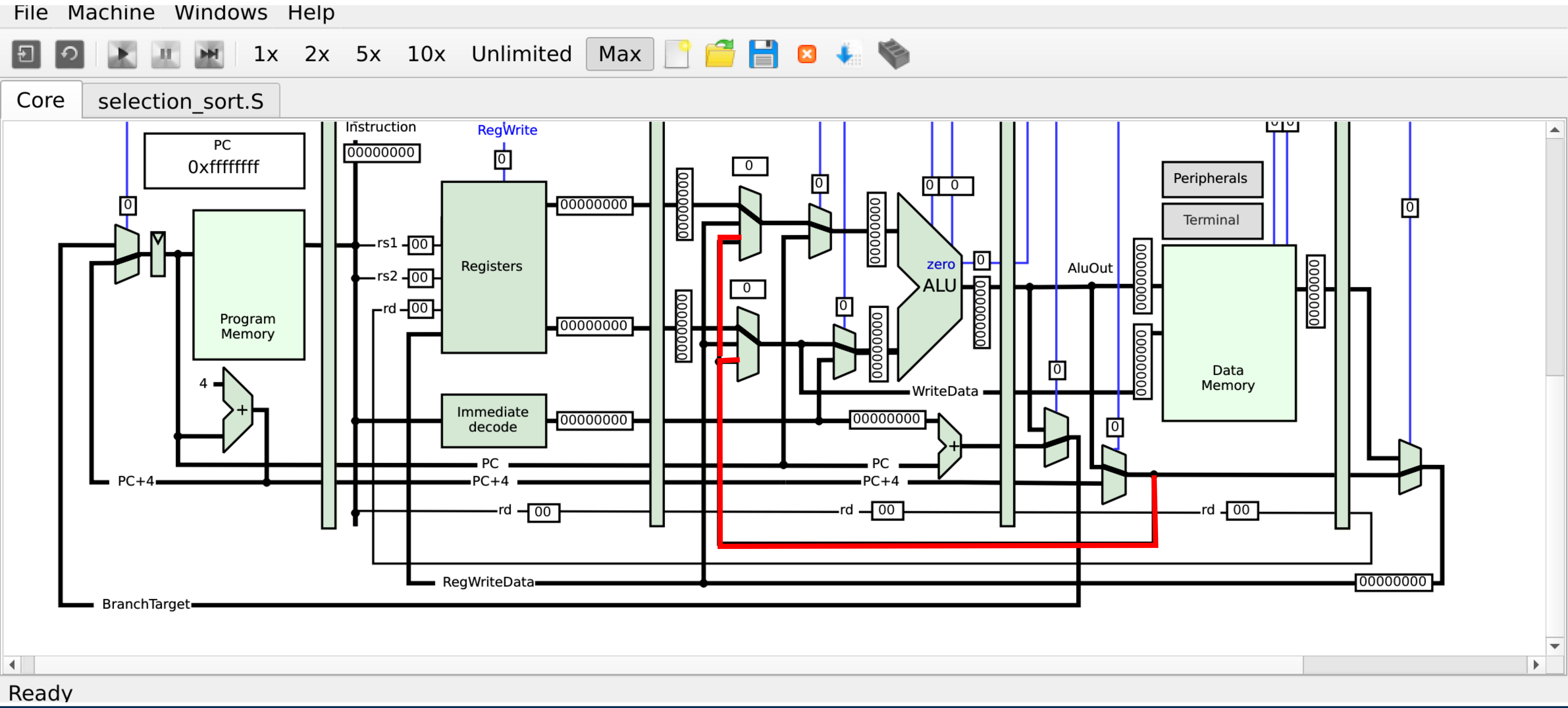
# Forwarding Paths



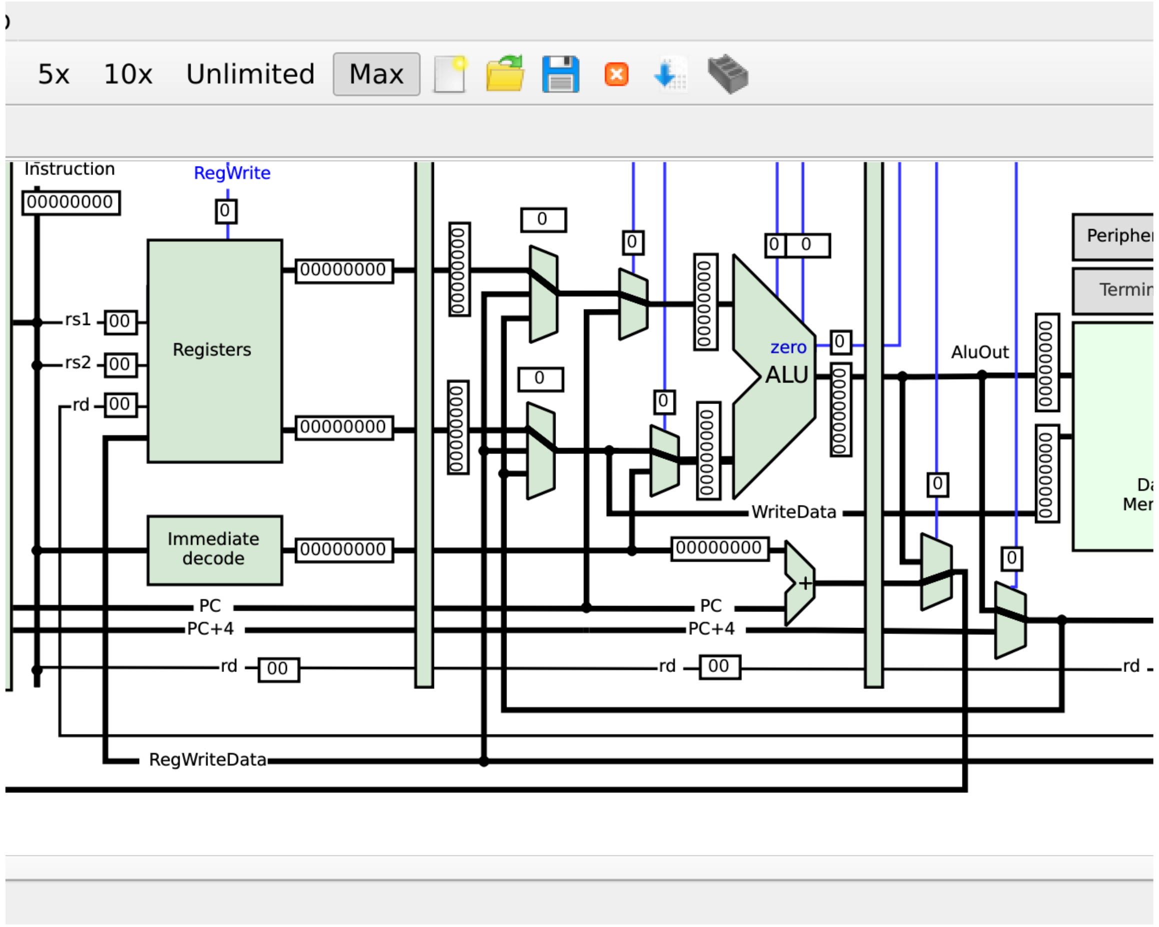
# Forwarding Paths



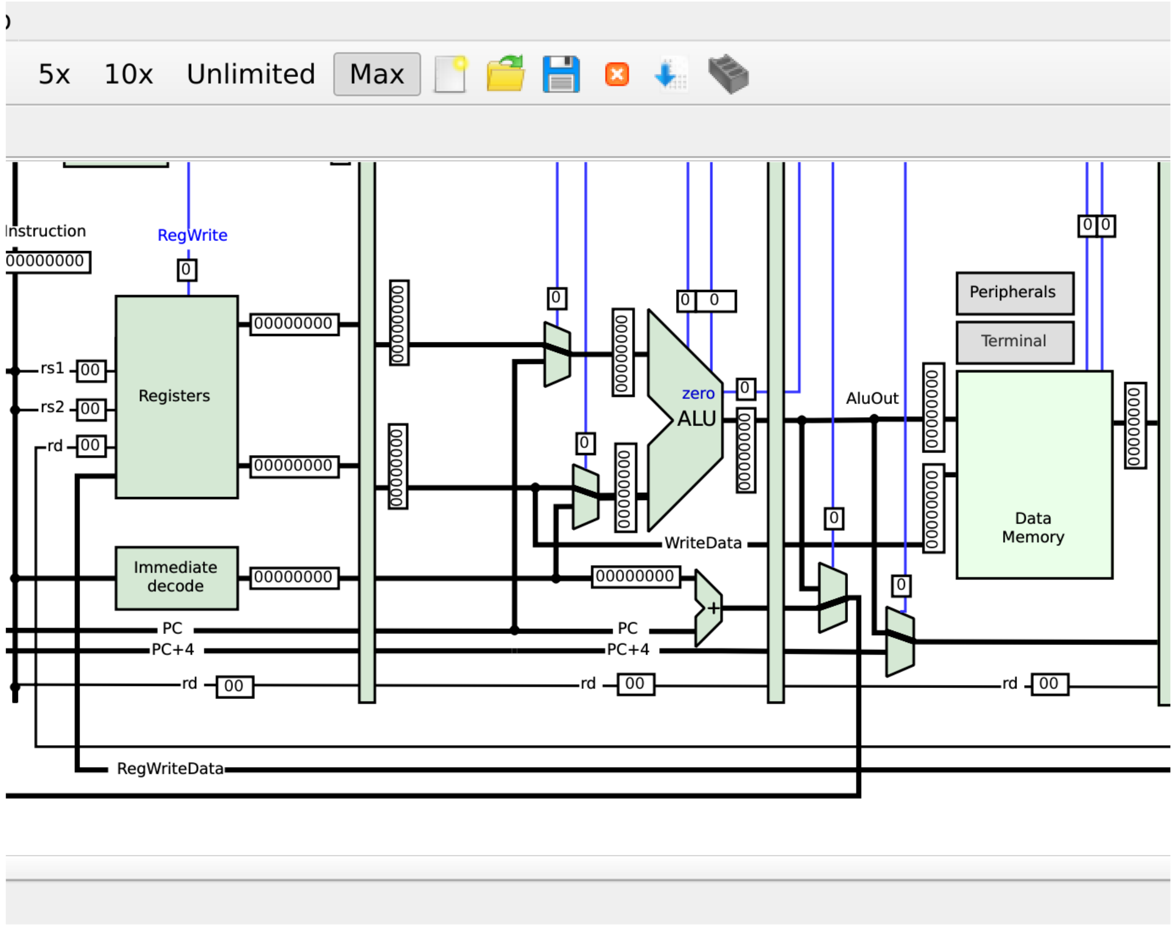
# Forwarding Paths



# Forward vs Stall

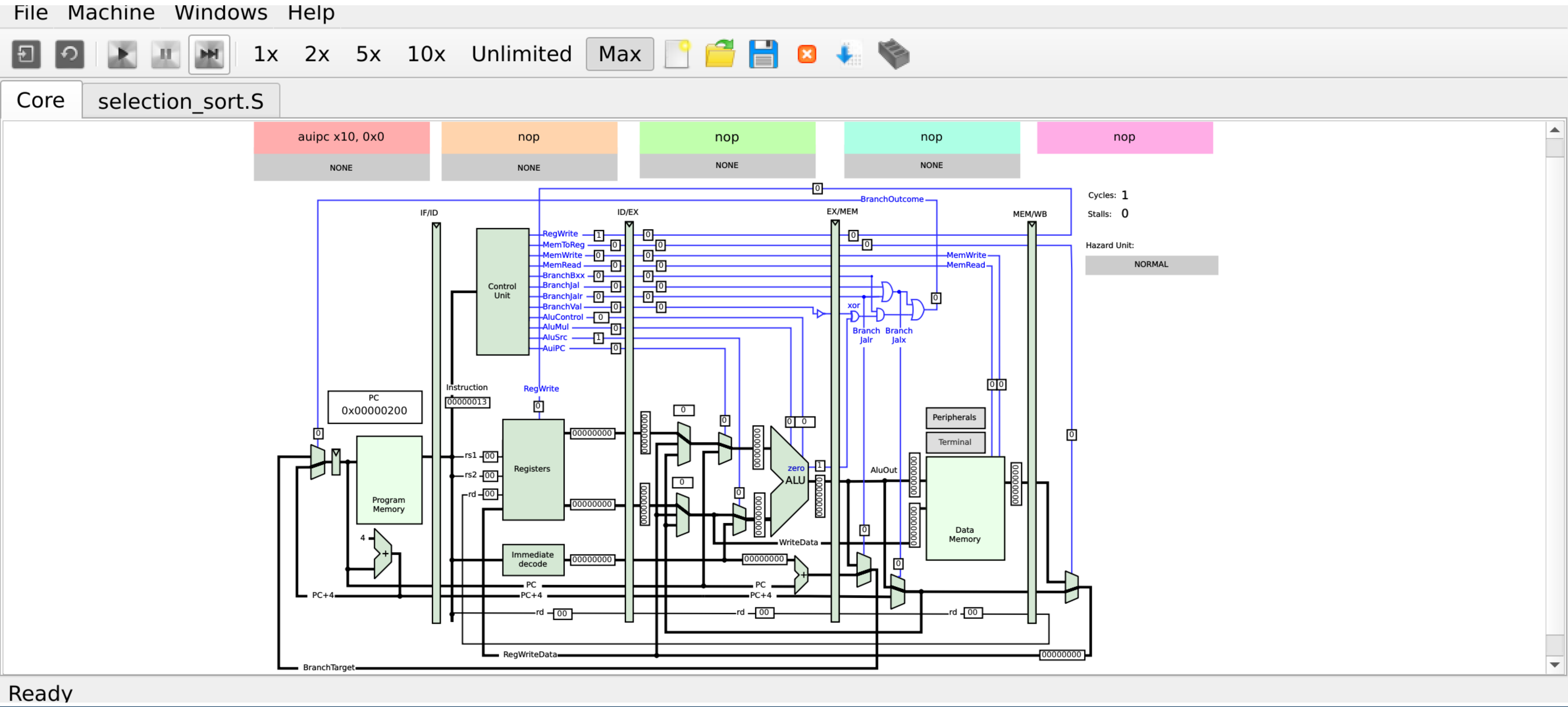


Forward



Stall Only

# Data Hazard Example 1



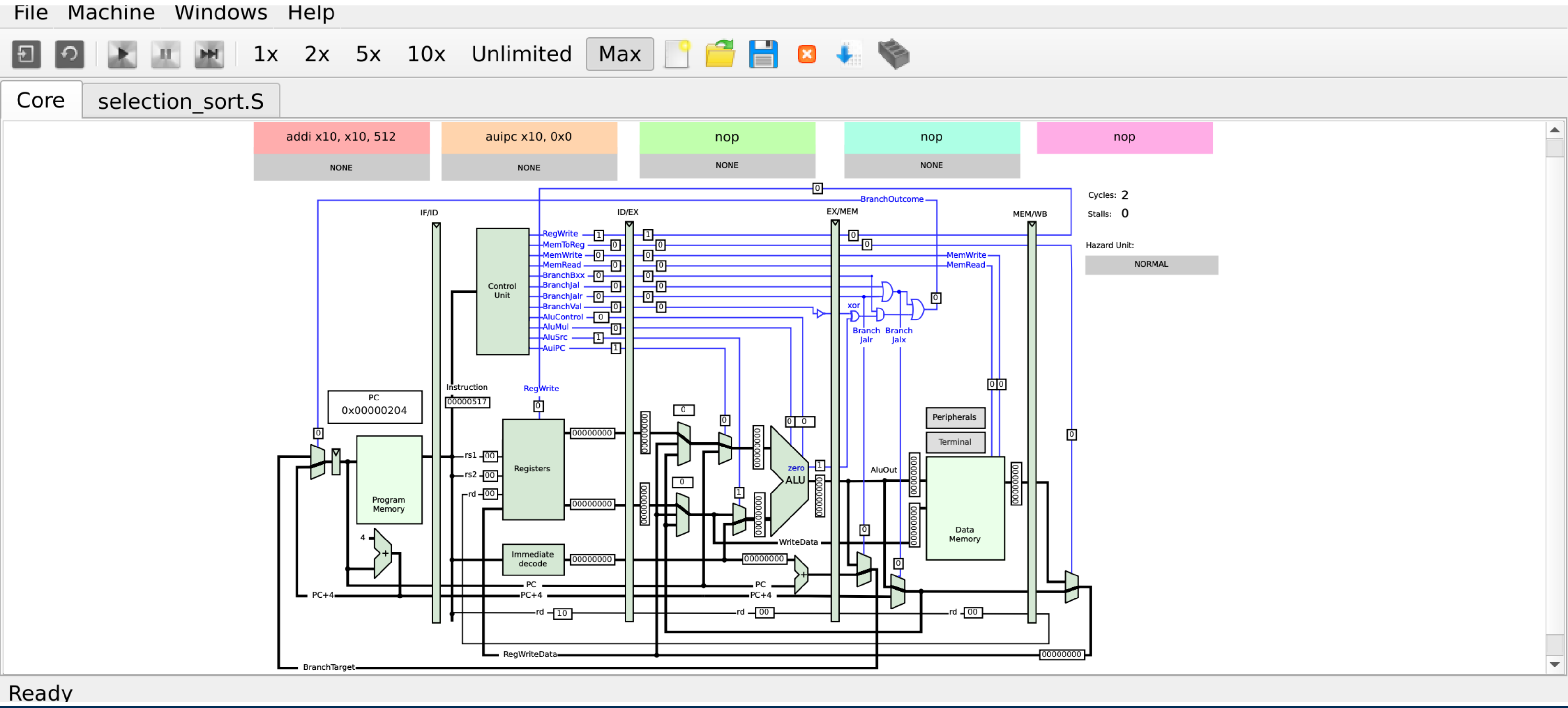
| auipc x10, 0x0 | nop | nop | nop | nop |
|---|---|---|---|---|
| NONE | NONE | NONE | NONE | |

PC 0x00000200

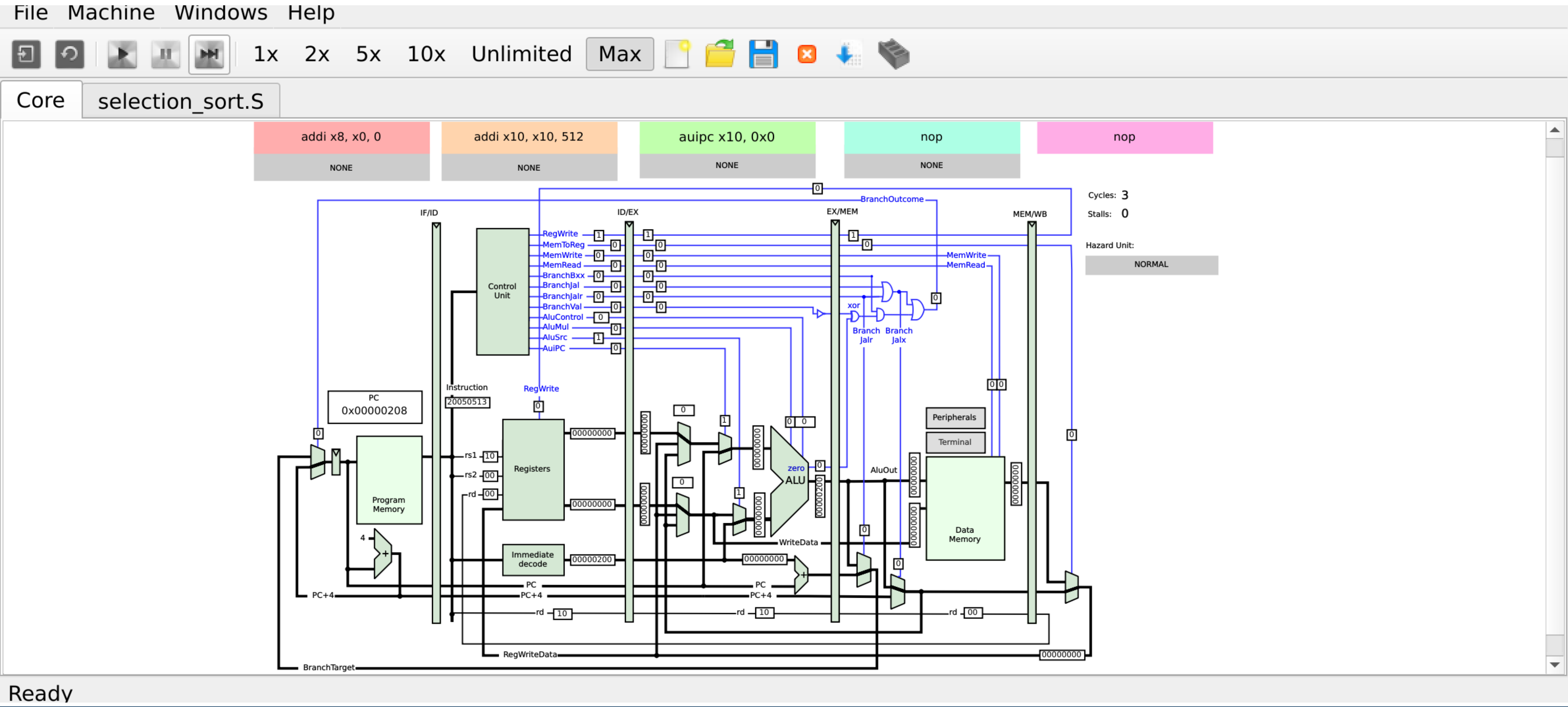
Cycles: 1
Stalls: 0
Hazard Unit: NORMAL

Ready

# Data Hazard Example 2



# Data Hazard Example 3



| addi x8, x0, 0 | addi x10, x10, 512 | auipc x10, 0x0 | nop | nop |
|---|---|---|---|---|
| NONE | NONE | NONE | NONE | |

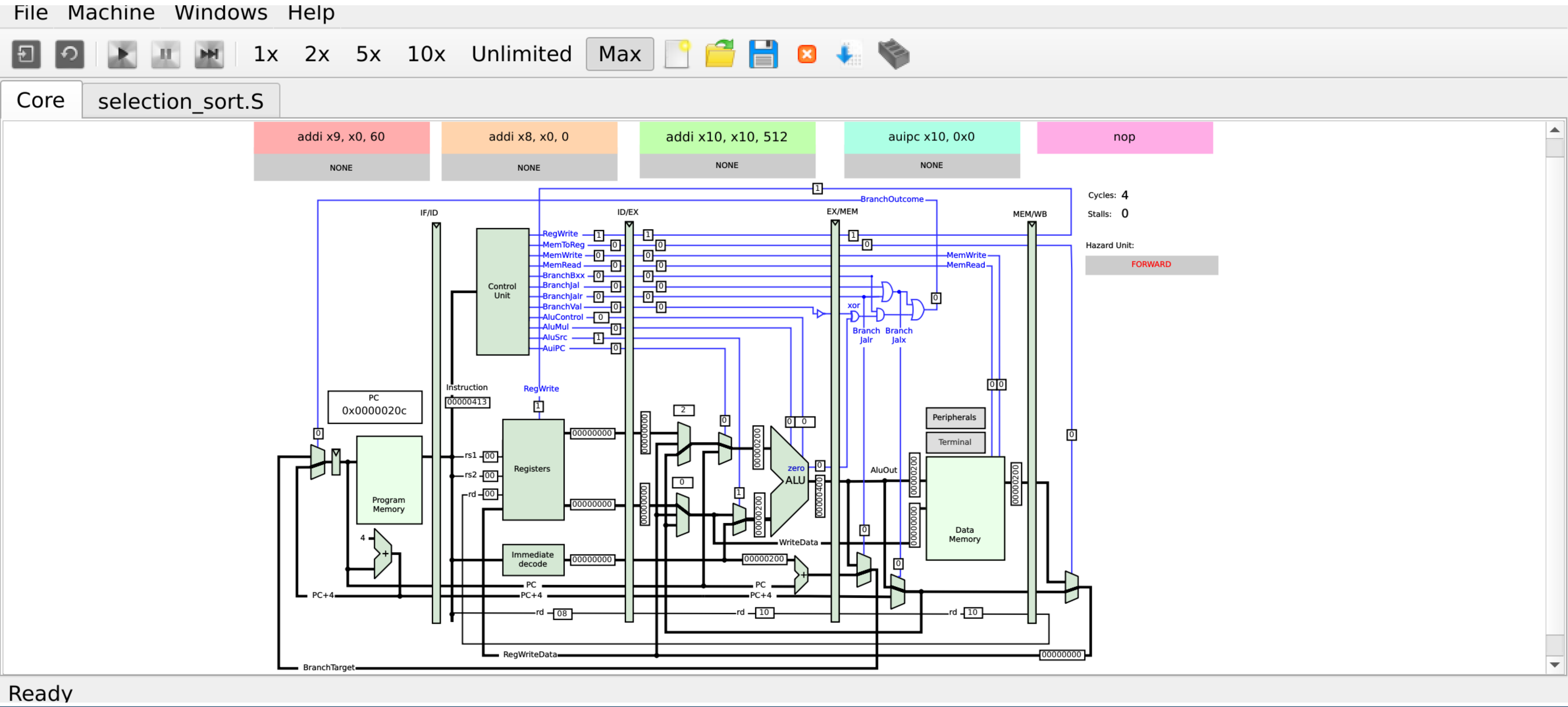
Cycles: 3

Stalls: 0

Hazard Unit: NORMAL

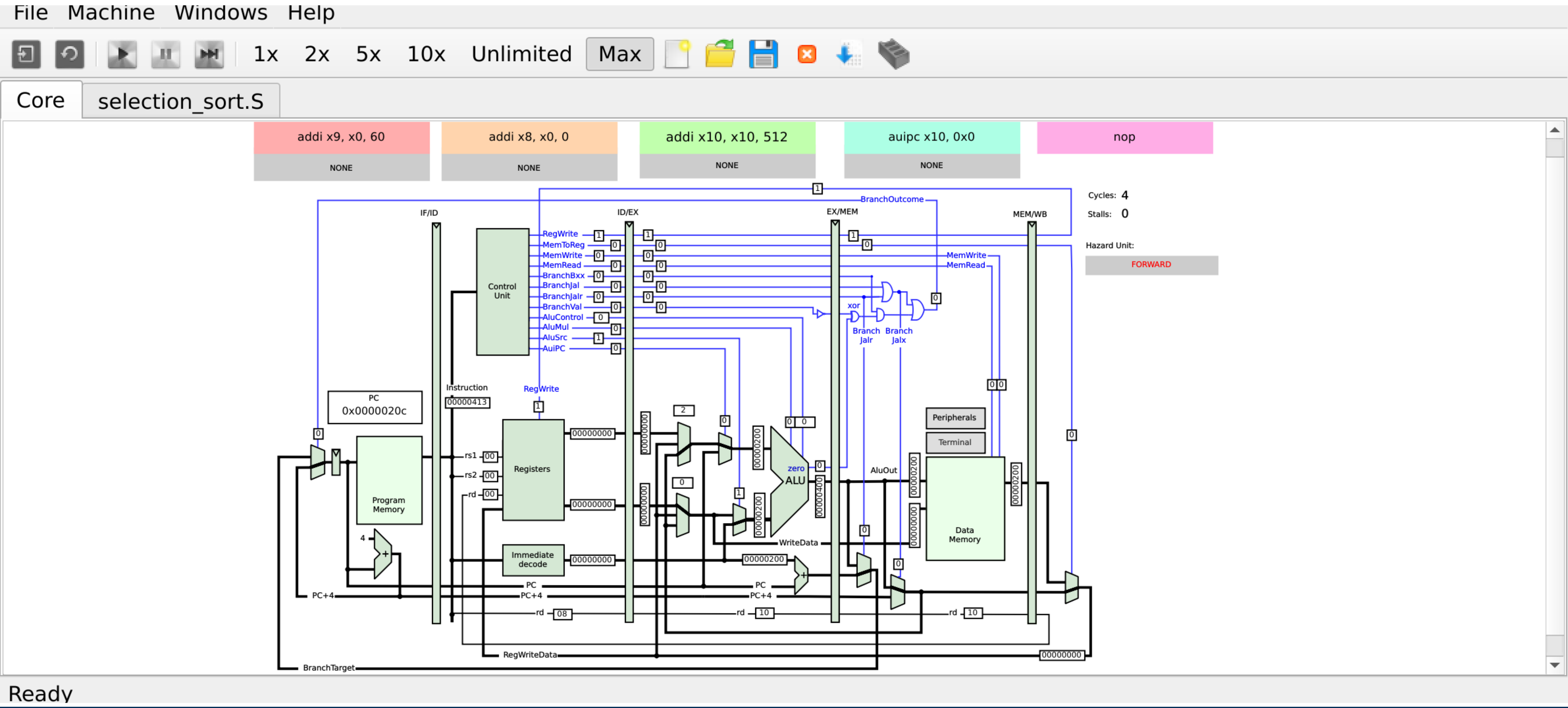
Ready

# Data Hazard Example: Forwarded



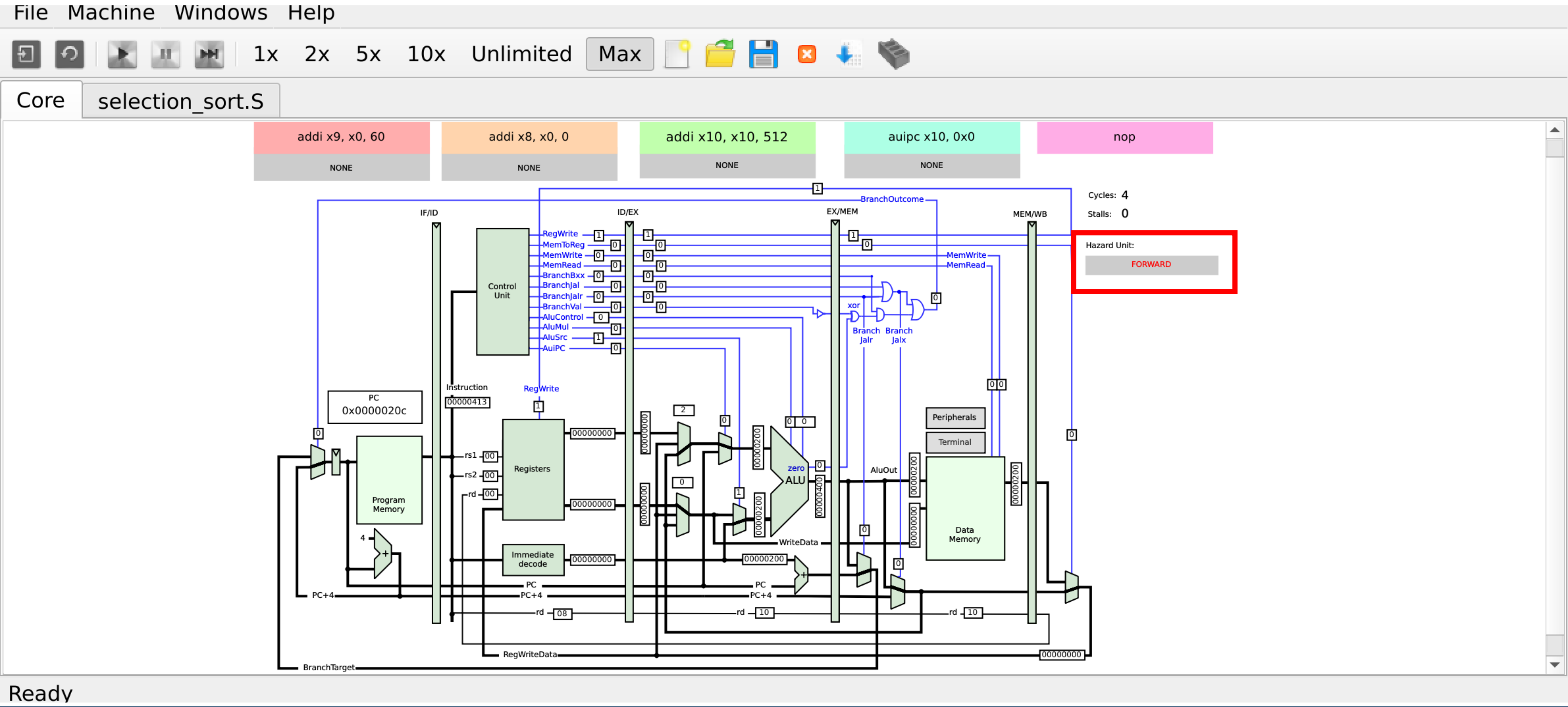
Cycles: 4
Stalls: 0
Hazard Unit:
FORWARD

# Data Hazard Example: Forwarded

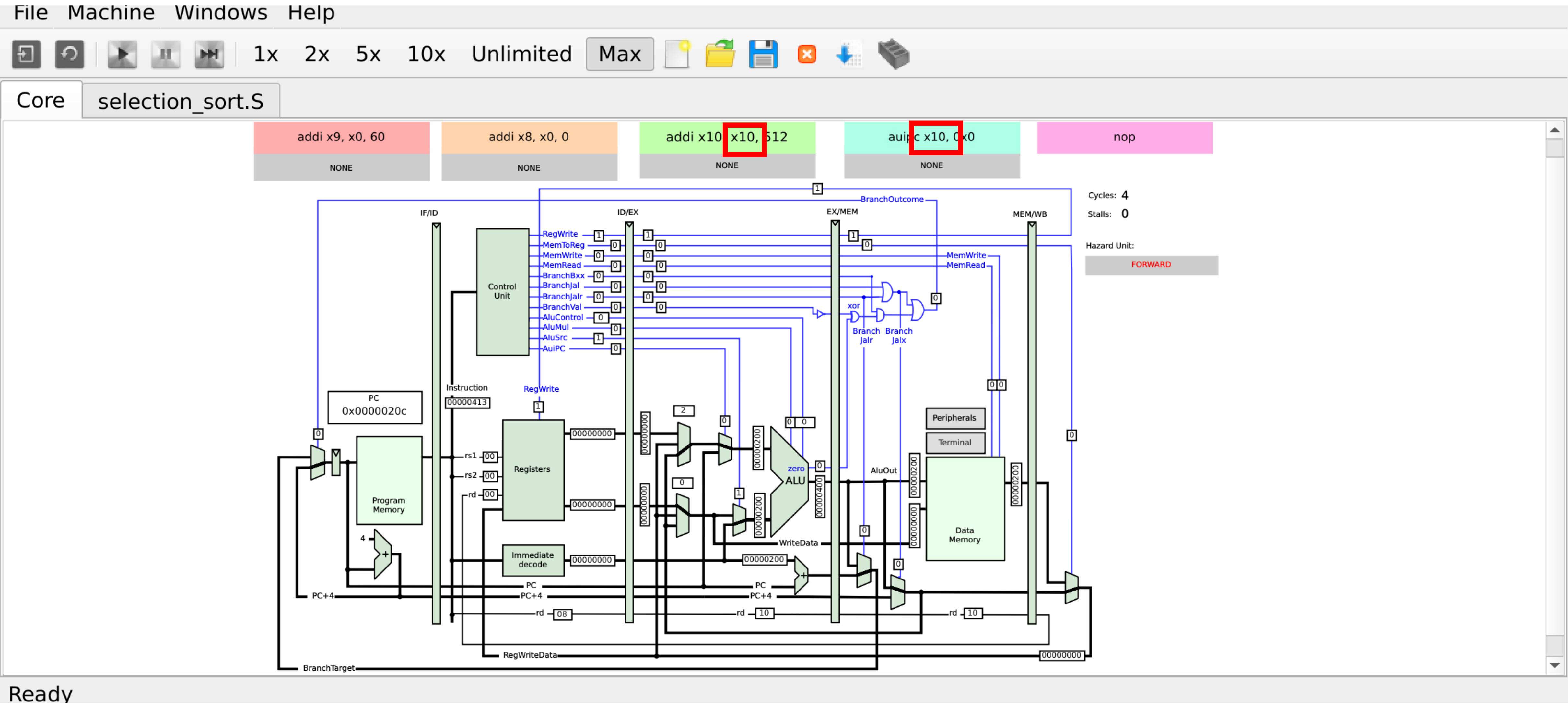


Cycles: 4
Stalls: 0
Hazard Unit: FORWARD

# Data Hazard Example: Forwarded



# Data Hazard Example: Forwarded



| addi x9, x0, 60 | addi x8, x0, 0 | addi x10, x10, 512 | auipc x10, 0x0 | nop |
|---|---|---|---|---|
| NONE | NONE | NONE | NONE | |

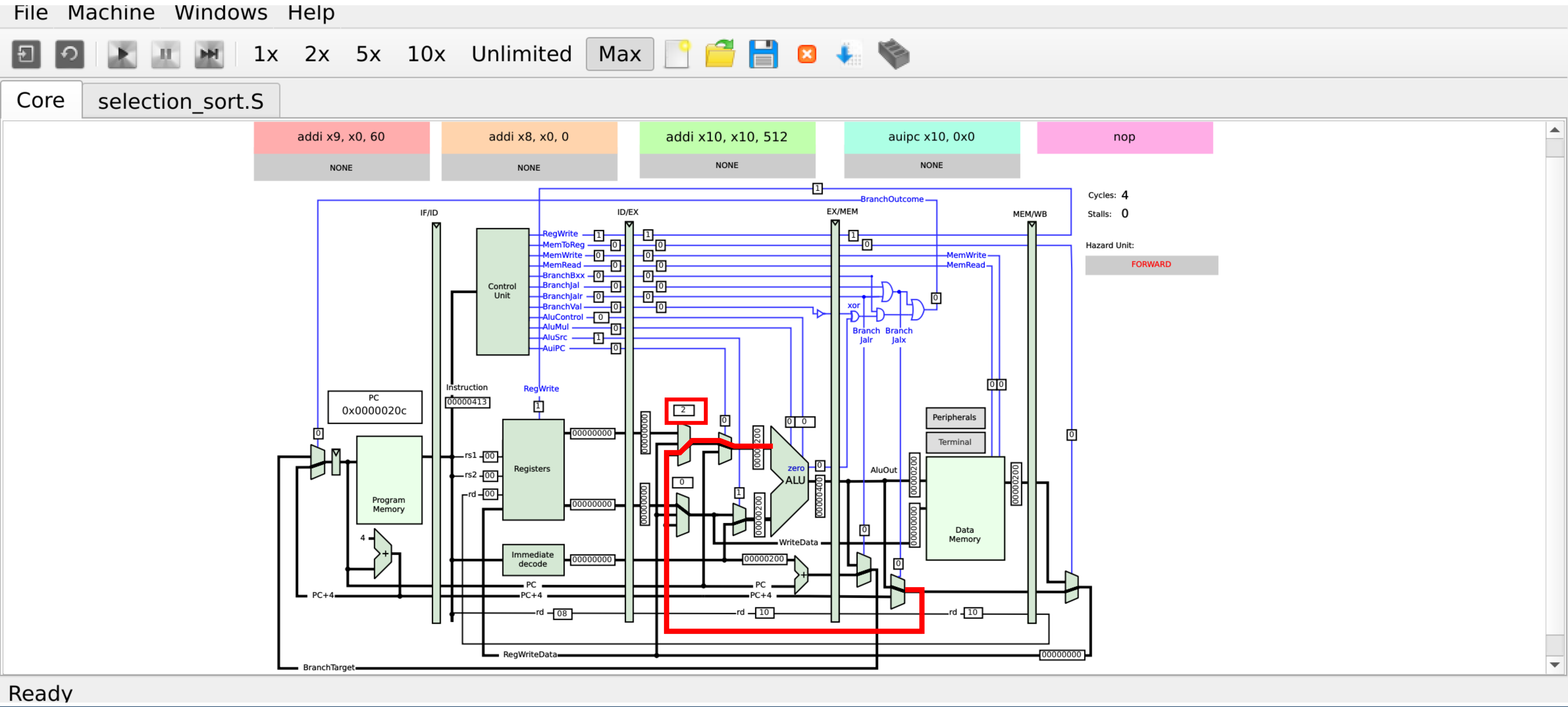
Cycles: 4

Stalls: 0

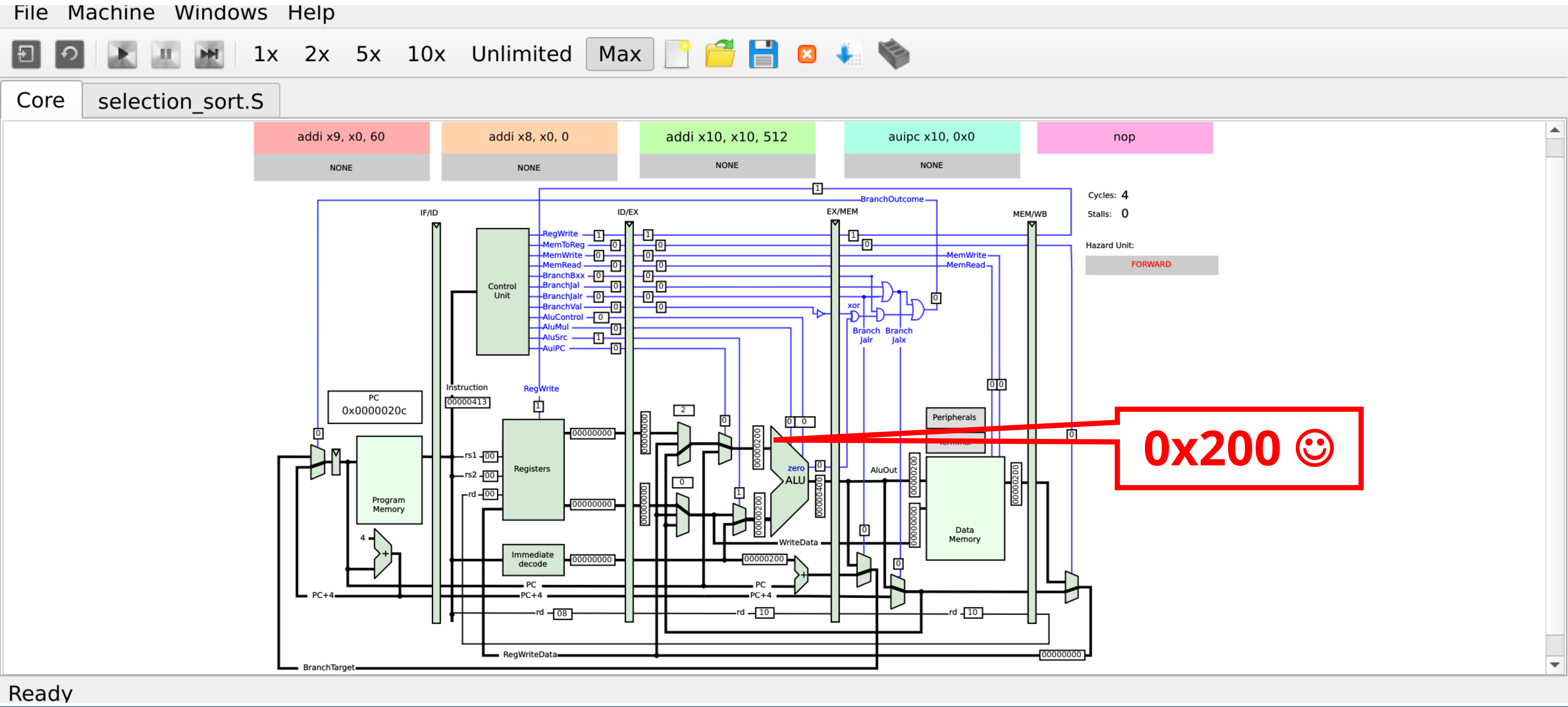
Hazard Unit:

FORWARD

# Data Hazard Example: Forwarded

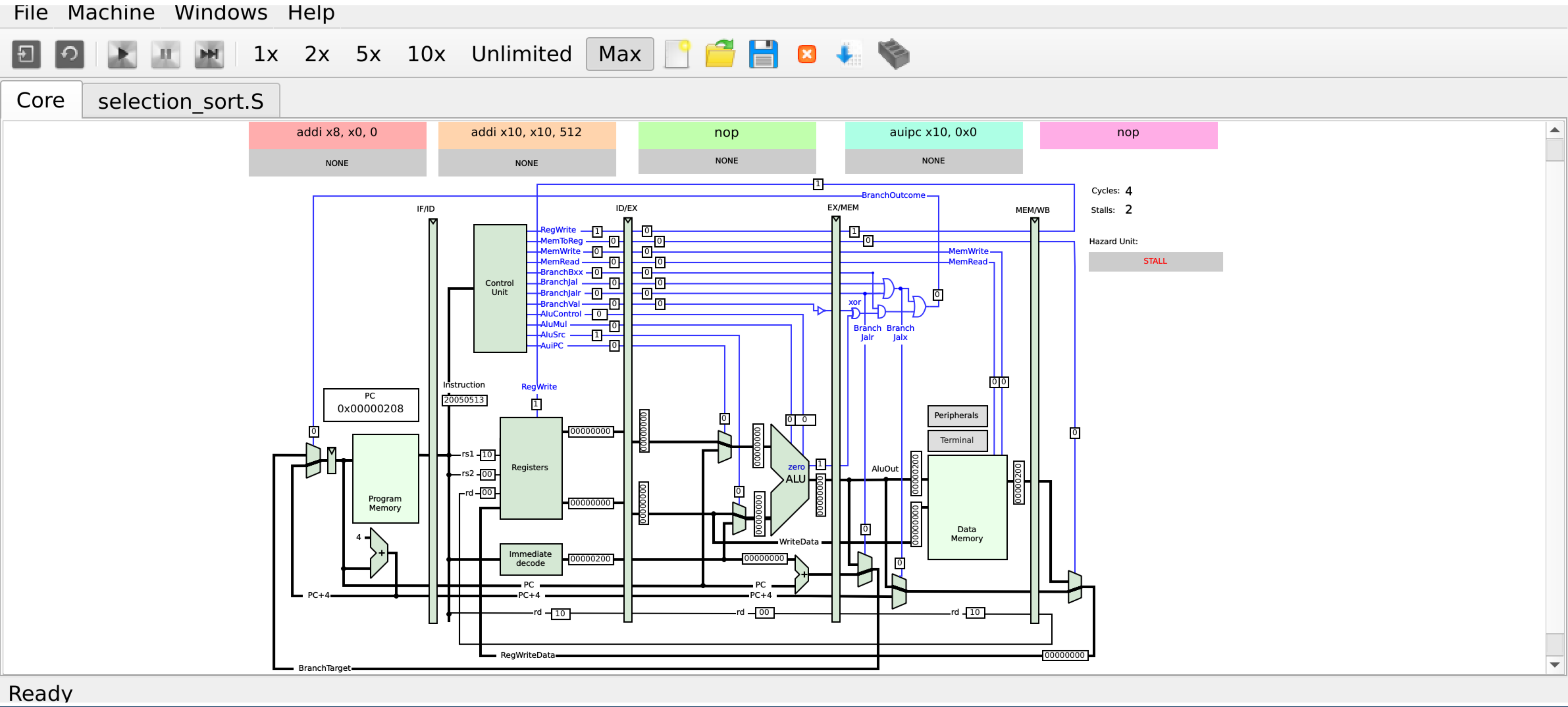


# Data Hazard Example: Forwarded



0x200 ☺

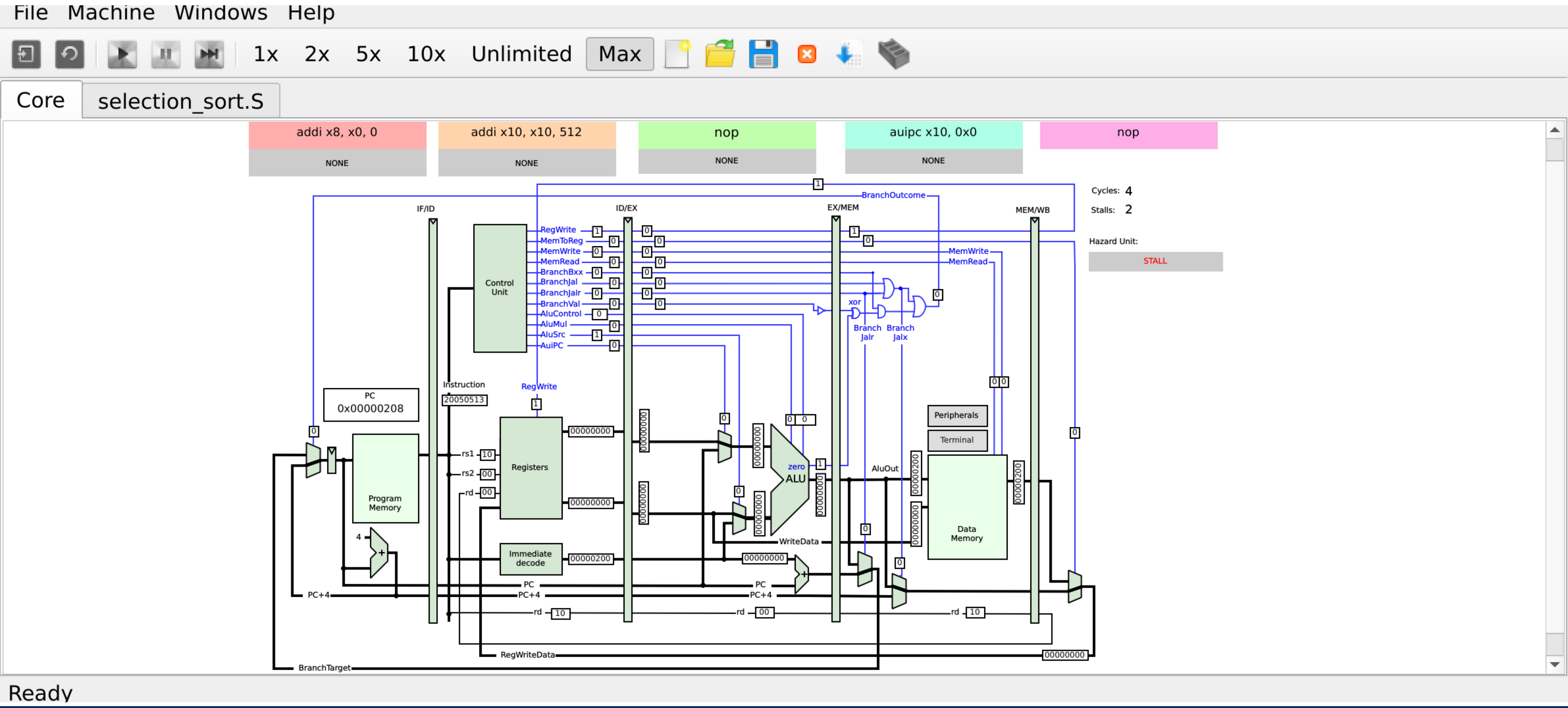
# Data Hazard Example: Stalled 1



| addi x8, x0, 0 | addi x10, x10, 512 | nop | auipc x10, 0x0 | nop |
|---|---|---|---|---|
| NONE | NONE | NONE | NONE | |

Cycles: 4
Stalls: 2
Hazard Unit:
STALL

# Data Hazard Example: Stalled 1



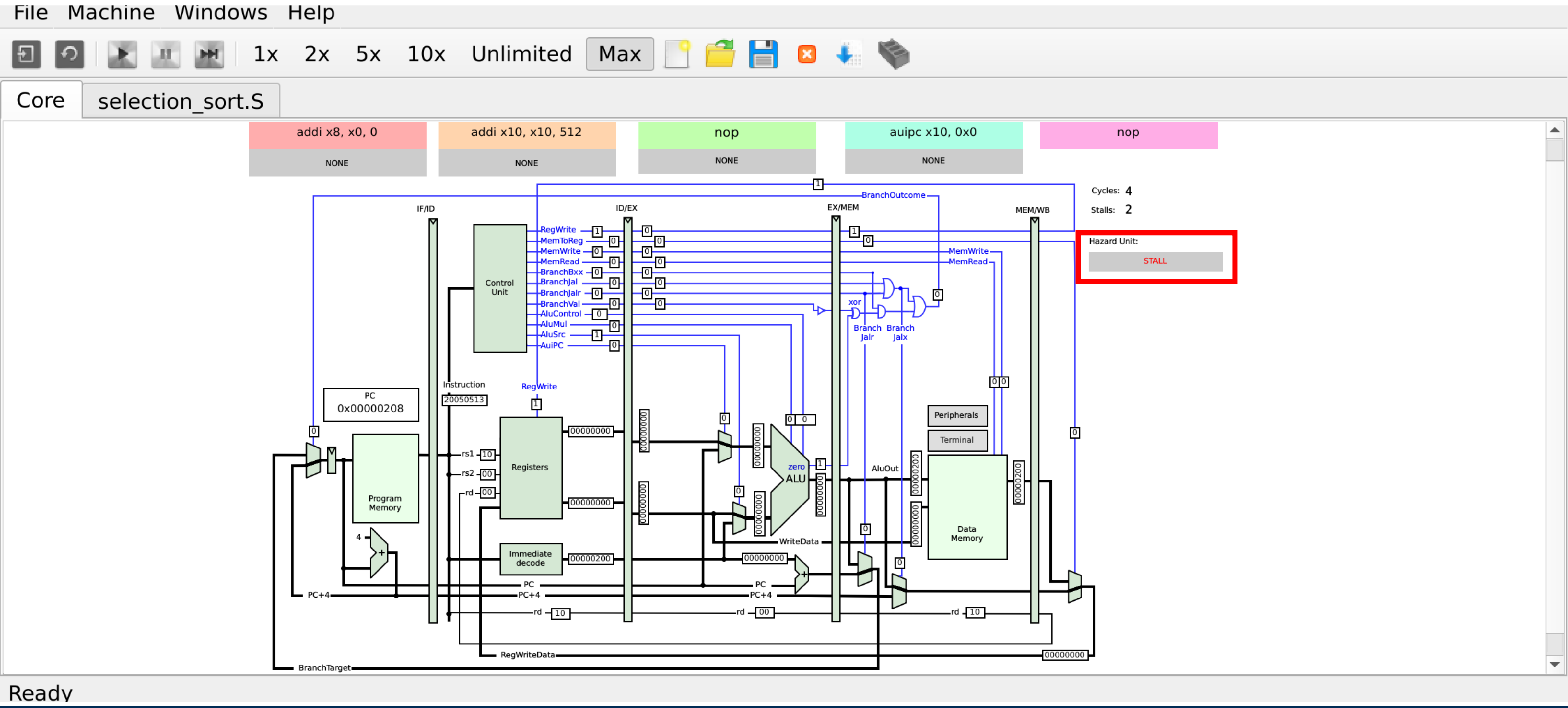
| addi x8, x0, 0 | addi x10, x10, 512 | nop | auipc x10, 0x0 | nop |
|---|---|---|---|---|
| NONE | NONE | NONE | NONE | |

Cycles: 4
Stalls: 2

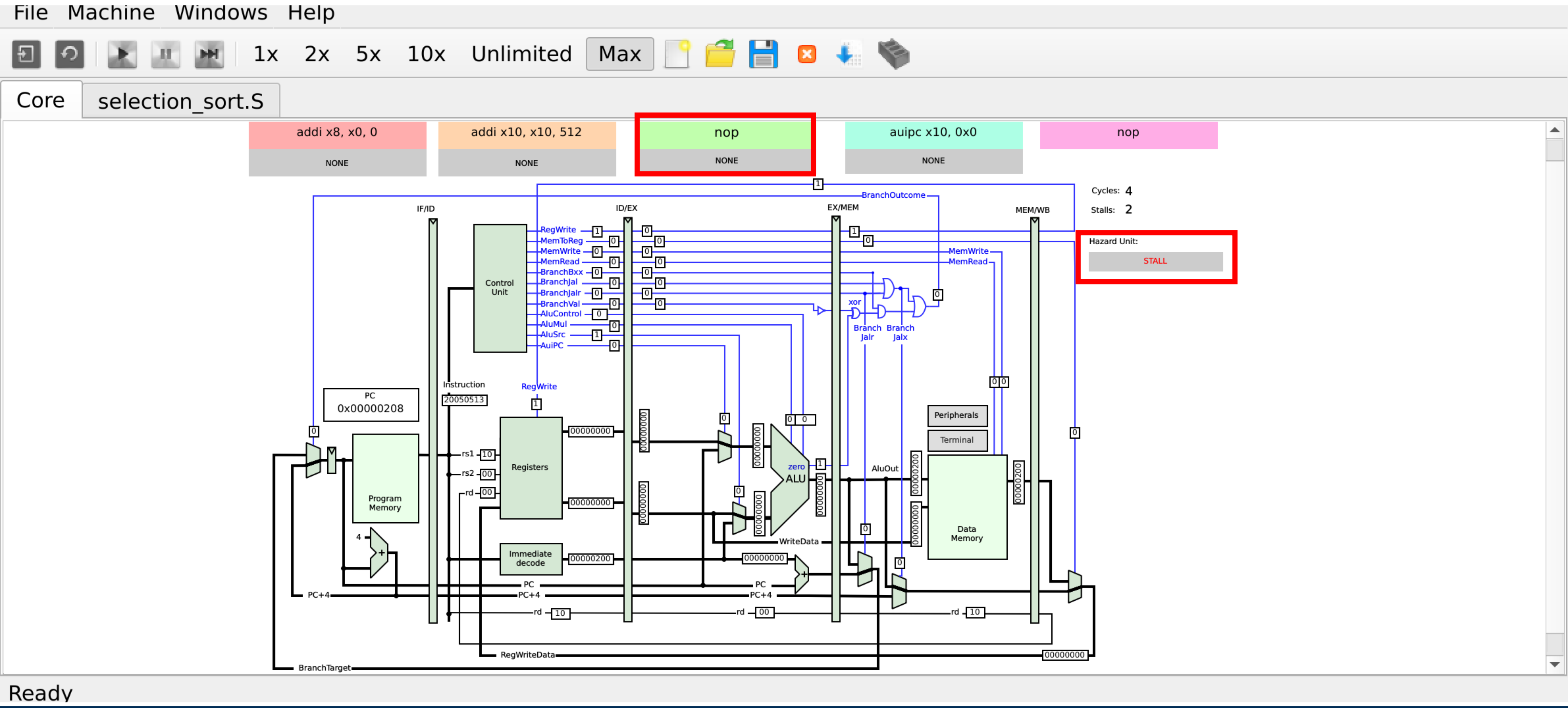
Hazard Unit:
STALL

Ready

# Data Hazard Example: Stalled 1



# Data Hazard Example: Stalled 1



File Machine Windows Help

1x 2x 5x 10x Unlimited Max

Core selection_sort.S

| addi x8, x0, 0 | addi x10, x10, 512 | nop | auipc x10, 0x0 | nop |
|---|---|---|---|---|
| NONE | NONE | NONE | NONE | |

Cycles: 4
Stalls: 2

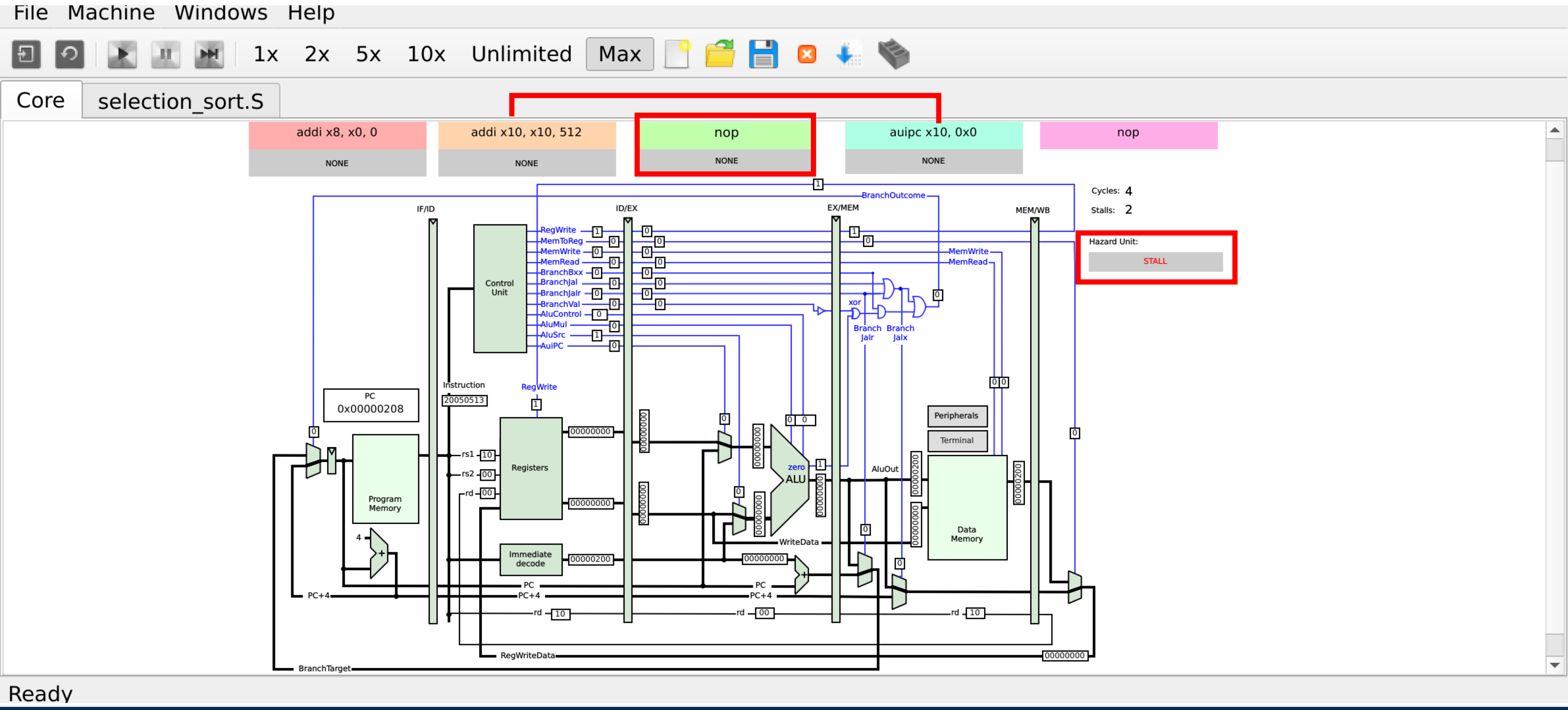
Hazard Unit: STALL

PC 0x00000208

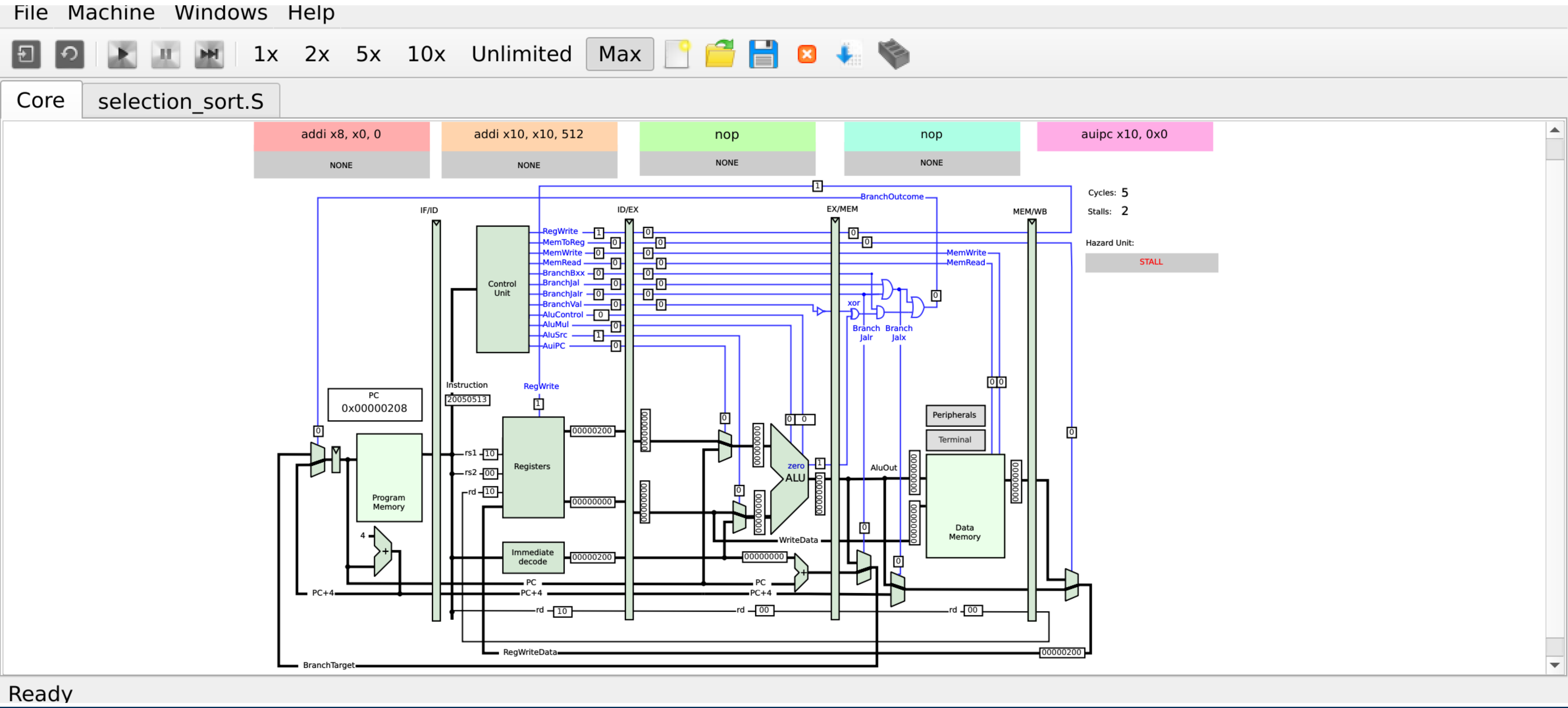
Instruction 20050513

Ready

# Data Hazard Example: Stalled 1



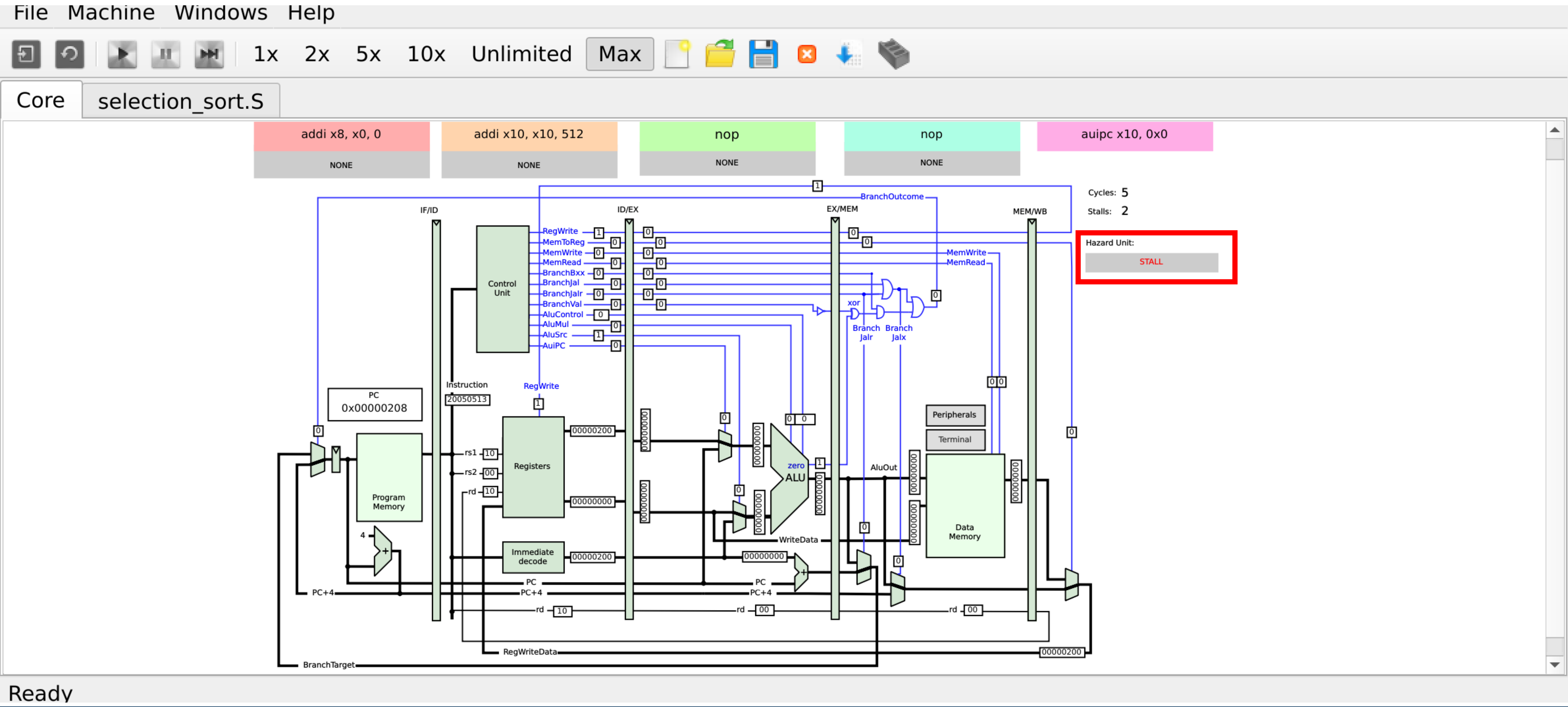
# Data Hazard Example: Stalled 2



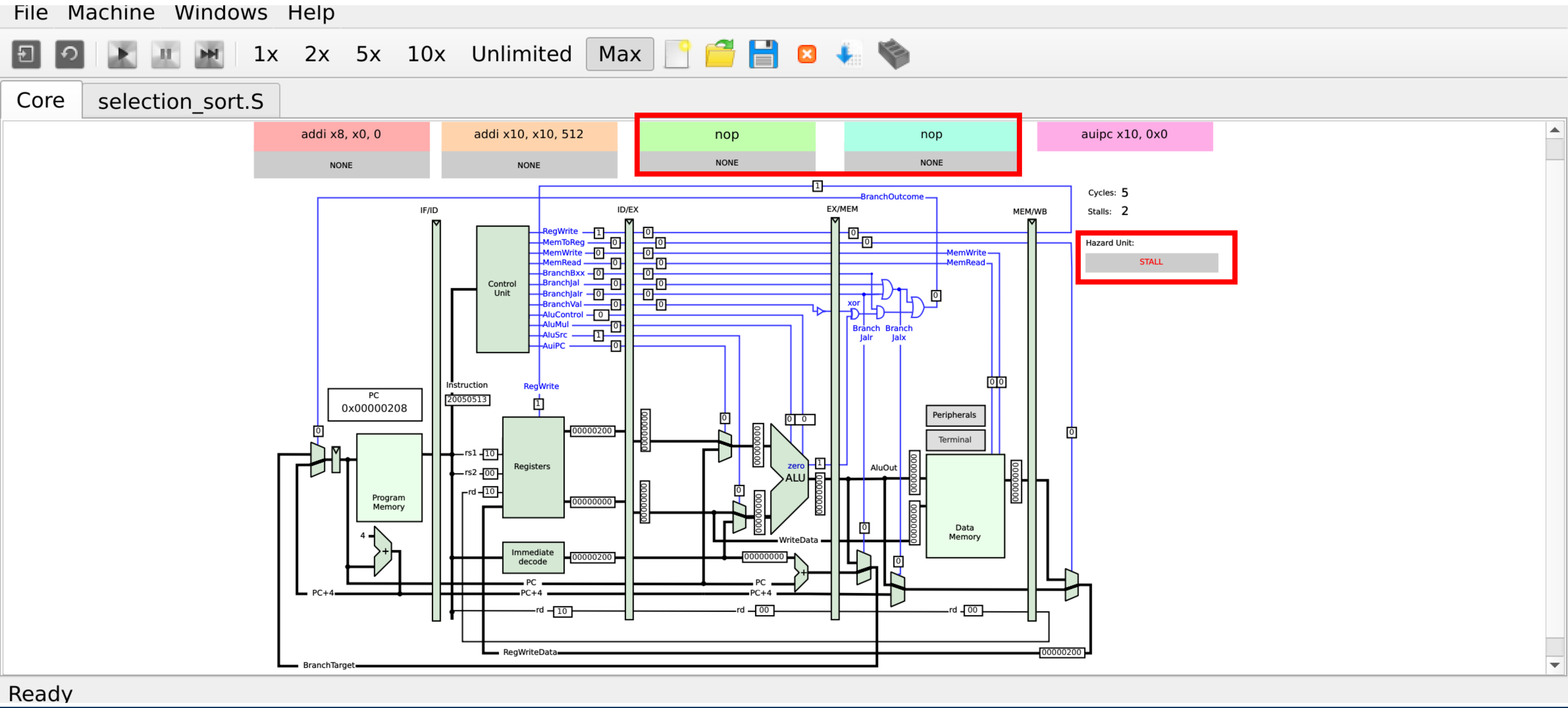
| addi x8, x0, 0 | addi x10, x10, 512 | nop | nop | auipc x10, 0x0 |
|---|---|---|---|---|
| NONE | NONE | NONE | NONE | |

Cycles: 5
Stalls: 2

# Data Hazard Example: Stalled 2



# Data Hazard Example: Stalled 2



addi x8, x0, 0 — NONE | addi x10, x10, 512 — NONE | nop — NONE | nop — NONE | auipc x10, 0x0

PC 0x00000208

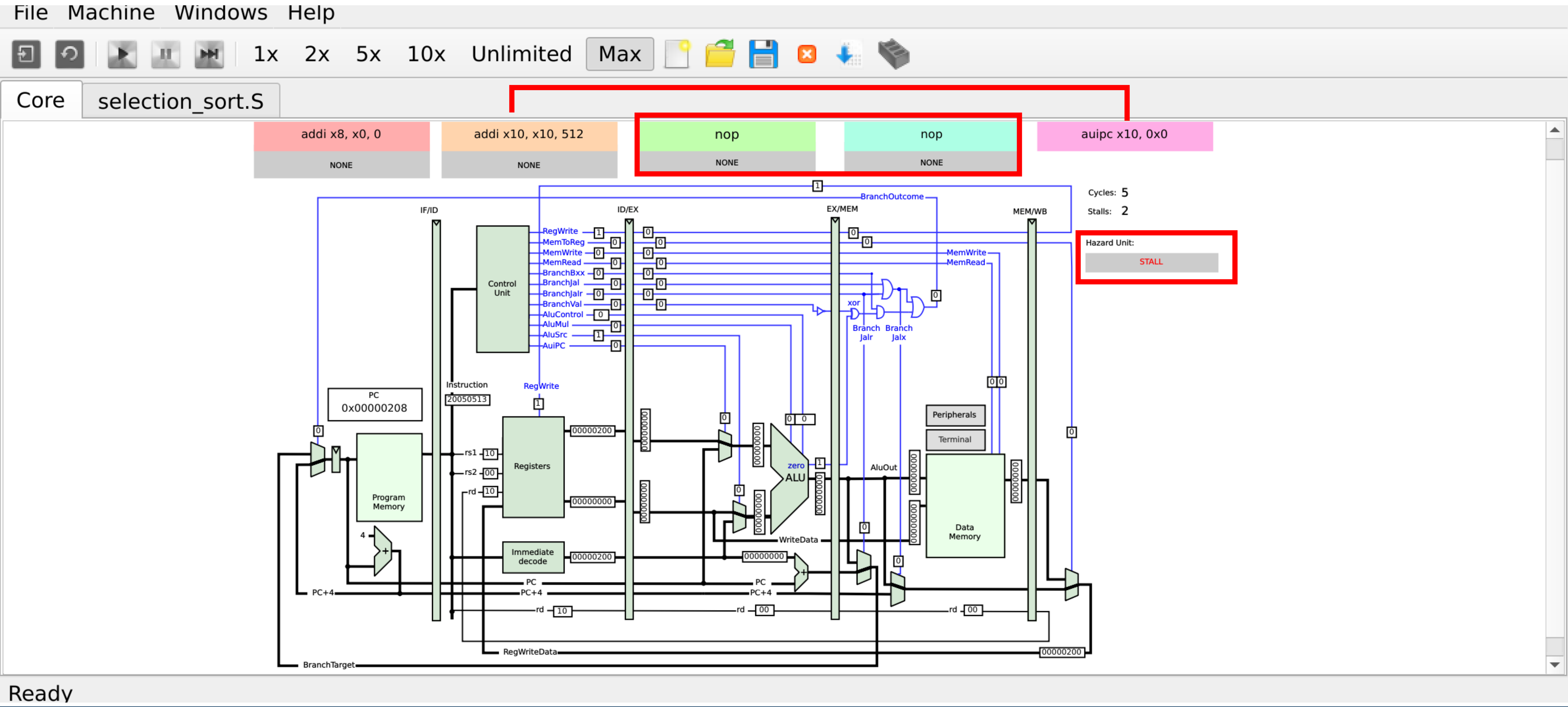
Instruction 20050513

Cycles: 5
Stalls: 2

Hazard Unit: STALL

Ready

# Data Hazard Example: Stalled 2



| addi x8, x0, 0 | addi x10, x10, 512 | nop | nop | auipc x10, 0x0 |
|---|---|---|---|---|
| NONE | NONE | NONE | NONE | |

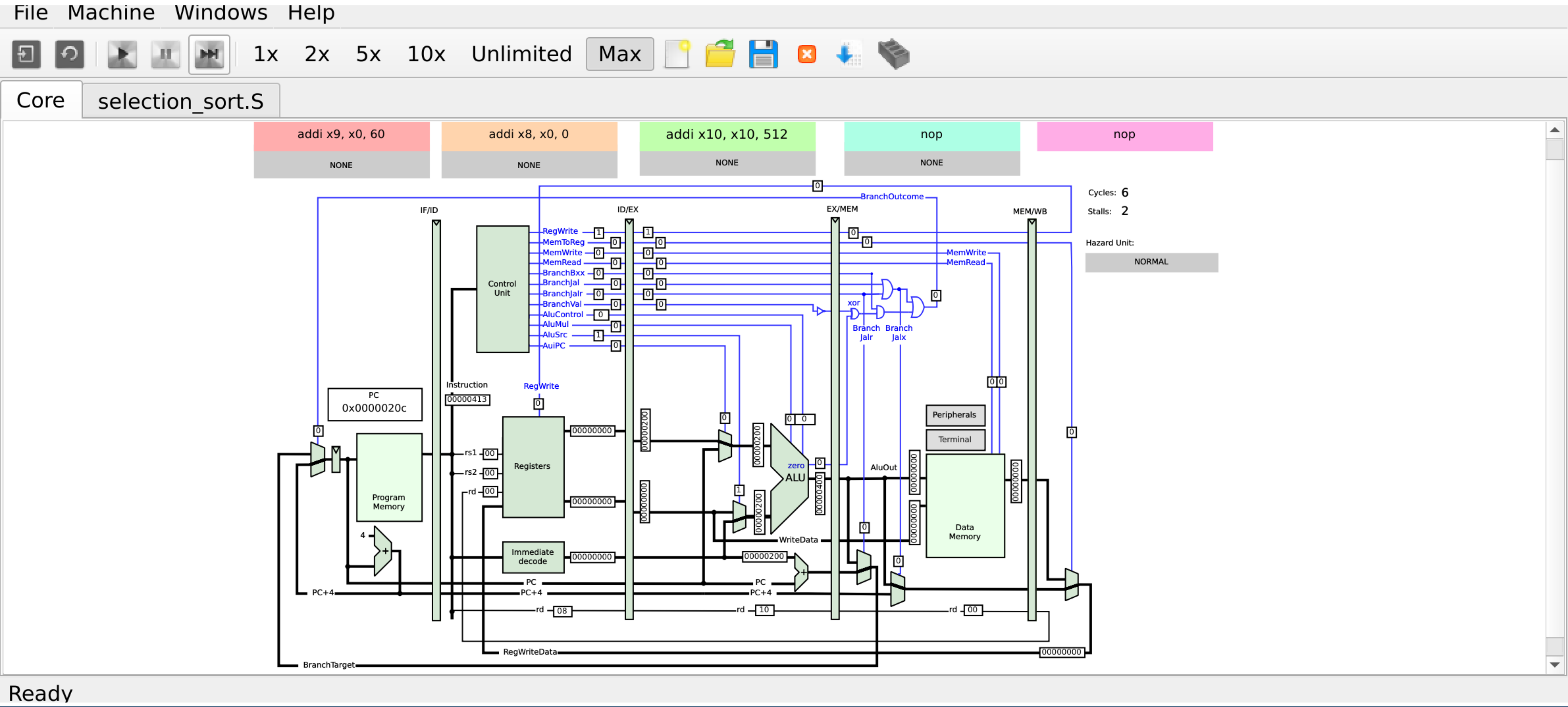
Cycles: 5
Stalls: 2

Hazard Unit: STALL

PC 0x00000208

Instruction 20050513

# Data Hazard Example: Stalled 3



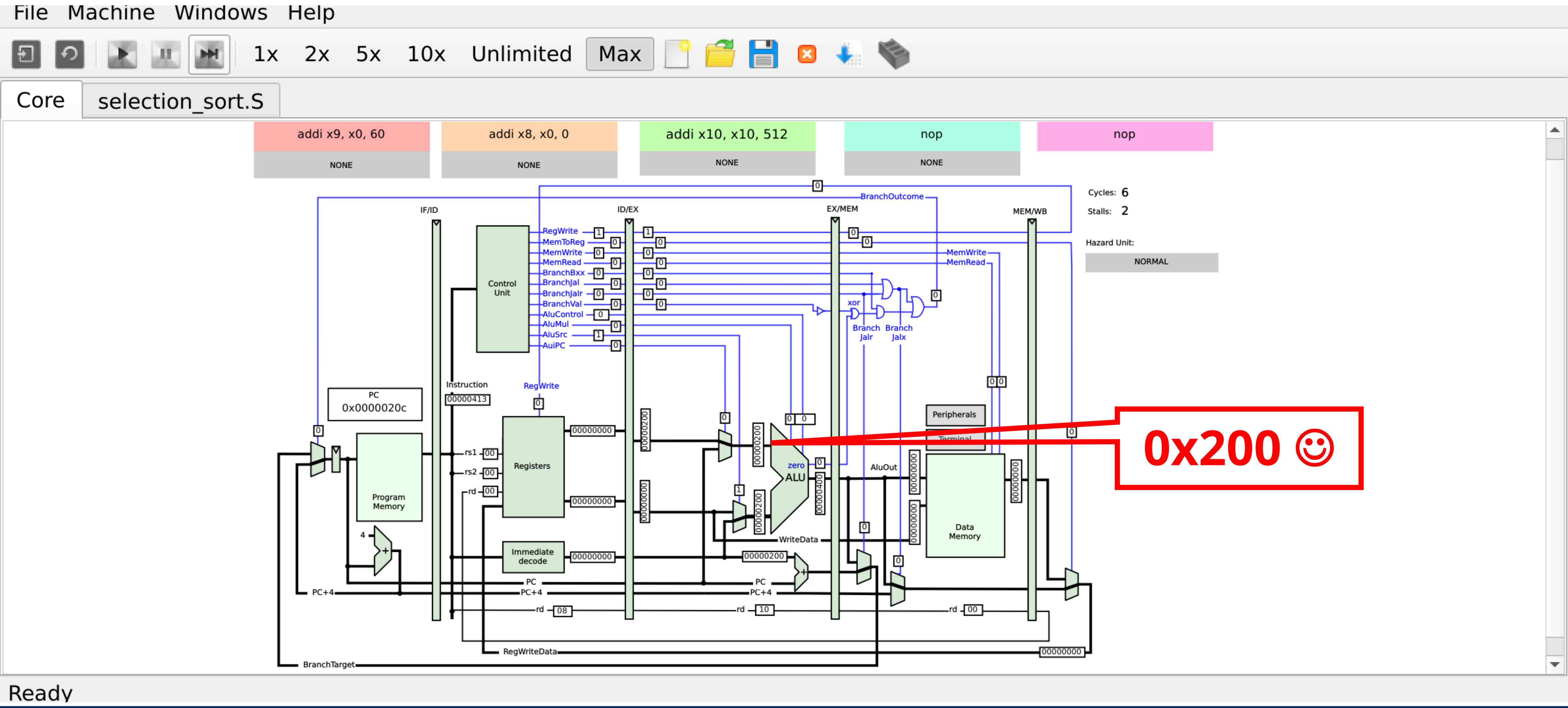
| addi x9, x0, 60 | addi x8, x0, 0 | addi x10, x10, 512 | nop | nop |
|---|---|---|---|---|
| NONE | NONE | NONE | NONE | |

Cycles: 6
Stalls: 2

Hazard Unit:
NORMAL

Ready

# Data Hazard Example: Stalled 3



| addi x9, x0, 60 | addi x8, x0, 0 | addi x10, x10, 512 | nop | nop |
|---|---|---|---|---|
| NONE | NONE | NONE | NONE | |

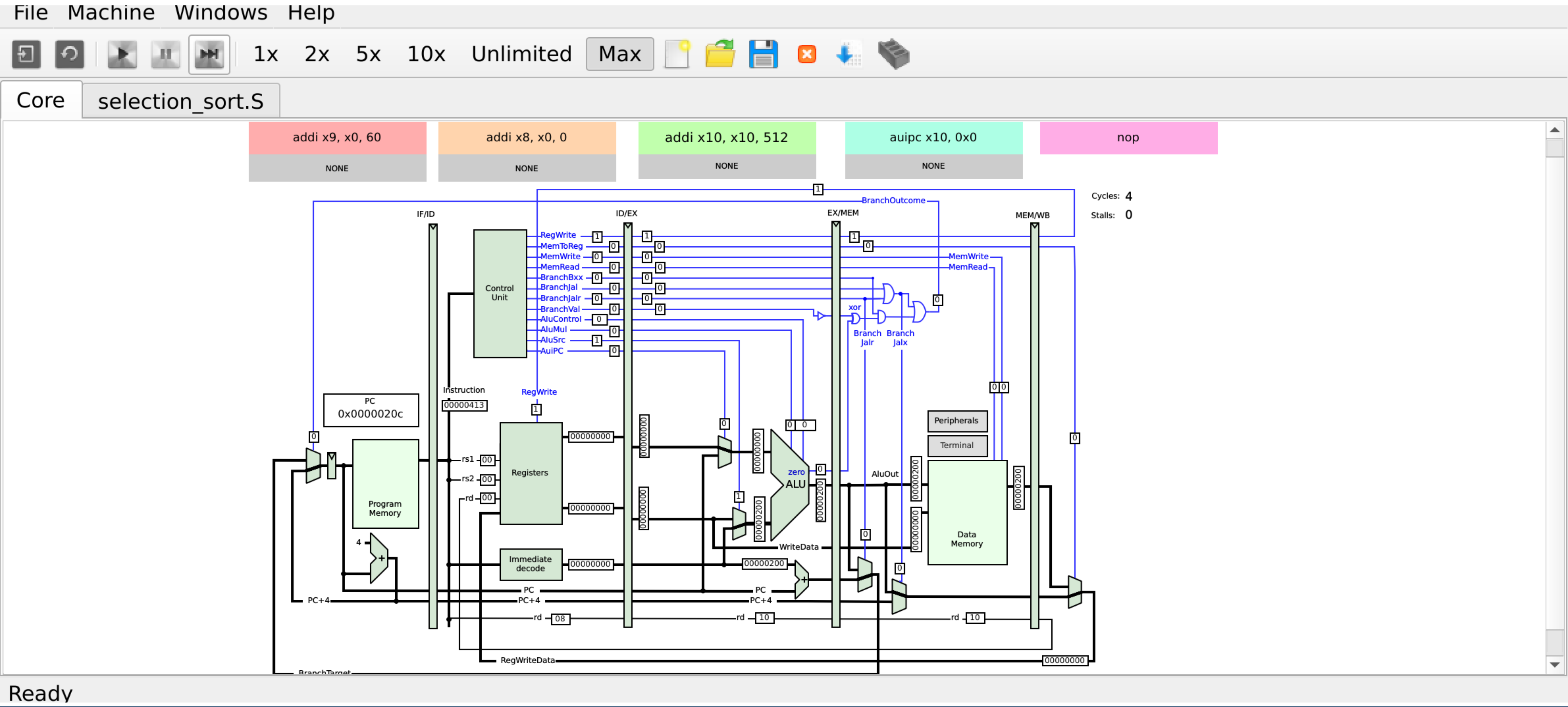
PC 0x0000020c

Instruction 00000413

Cycles: 6
Stalls: 2
Hazard Unit: NORMAL

0x200 ☺

# Data Hazard Example: Failed



| addi x9, x0, 60 | addi x8, x0, 0 | addi x10, x10, 512 | auipc x10, 0x0 | nop |
|---|---|---|---|---|
| NONE | NONE | NONE | NONE | |

PC 0x0000020c

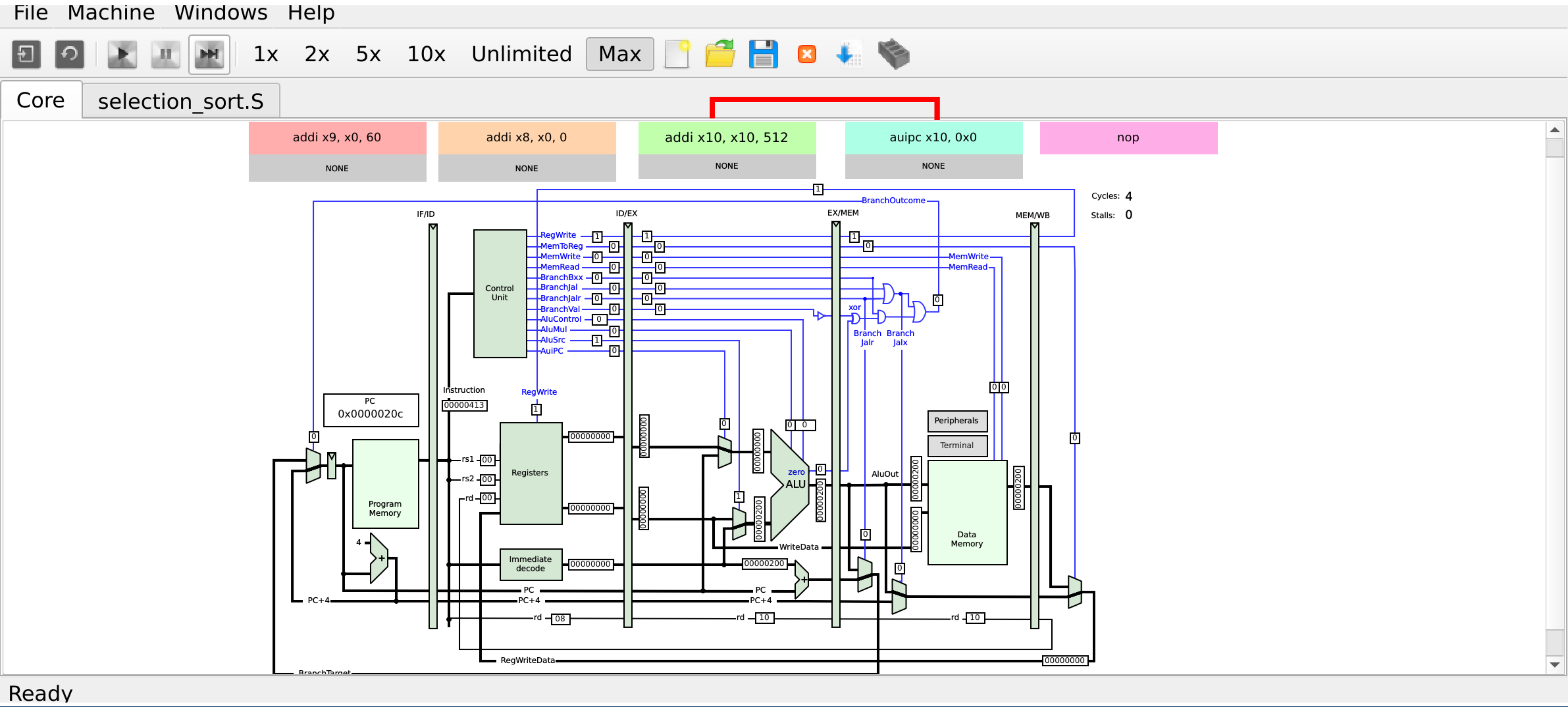
Instruction 00000413

Cycles: 4

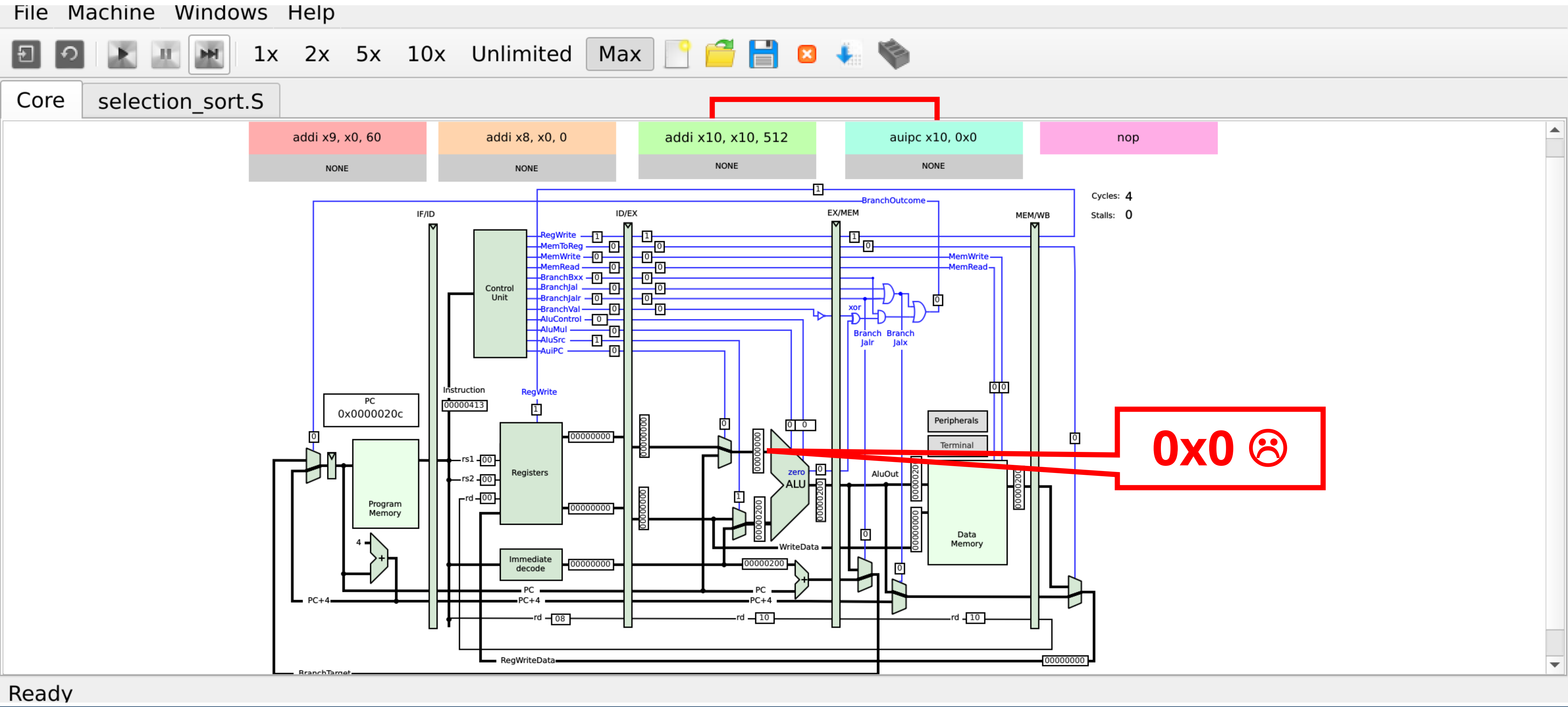
Stalls: 0

Ready

# Data Hazard Example: Failed



# Data Hazard Example: Failed



## Command Line Interface

Testing Automation
Quantitative Analysis

## CLI Example

```
>>> qtrvsim_cli --asm program.S --trace-fetch
```

# CLI Example

```
>>> qtrvsim_cli --asm program.S --trace-fetch

Fetch: addi x1, x0, 17

Fetch: addi x2, x0, 34

Fetch: lw x4, 68(x0)

Fetch: addi x5, x4, 85

Fetch: beq x0, x0, 0x22c

Fetch: addi x21, x0, 17

Machine stopped on BREAK exception.

Fetch: ebreak
```

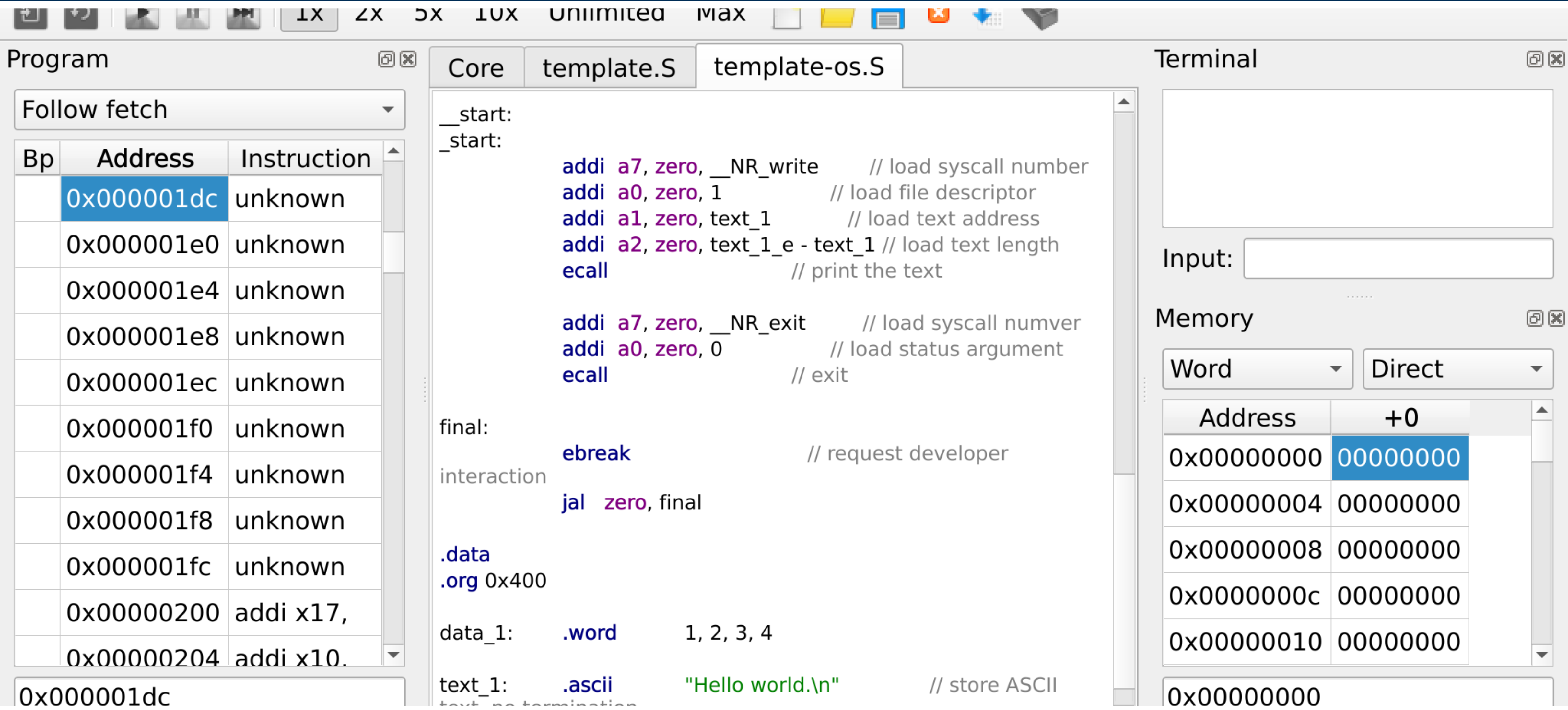
# CLI Capabilities

- Assembler
- Configuration
- Pipeline stage tracing
- Registers and memory tracing
- Register, memory and diagnostic data dump
- Memory load from file

# Peripherals and OS Emulation

Syscalls
LED
Knobs
LCD
Terminal
C programming

# OS Template



Program

Follow fetch

| Bp | Address | Instruction |
|---|---|---|
| | 0x000001dc | unknown |
| | 0x000001e0 | unknown |
| | 0x000001e4 | unknown |
| | 0x000001e8 | unknown |
| | 0x000001ec | unknown |
| | 0x000001f0 | unknown |
| | 0x000001f4 | unknown |
| | 0x000001f8 | unknown |
| | 0x000001fc | unknown |
| | 0x00000200 | addi x17, |
| | 0x00000204 | addi x10, |

0x000001dc

Core | template.S | template-os.S

```
__start:
_start:
            addi  a7, zero, __NR_write       // load syscall number
            addi  a0, zero, 1               // load file descriptor
            addi  a1, zero, text_1           // load text address
            addi  a2, zero, text_1_e - text_1 // load text length
            ecall                          // print the text

            addi  a7, zero, __NR_exit        // load syscall numver
            addi  a0, zero, 0               // load status argument
            ecall                          // exit

final:
            ebreak                          // request developer
interaction
            jal   zero, final

.data
.org 0x400

data_1:     .word       1, 2, 3, 4

text_1:     .ascii      "Hello world.\n"            // store ASCII
```

Terminal

Input:

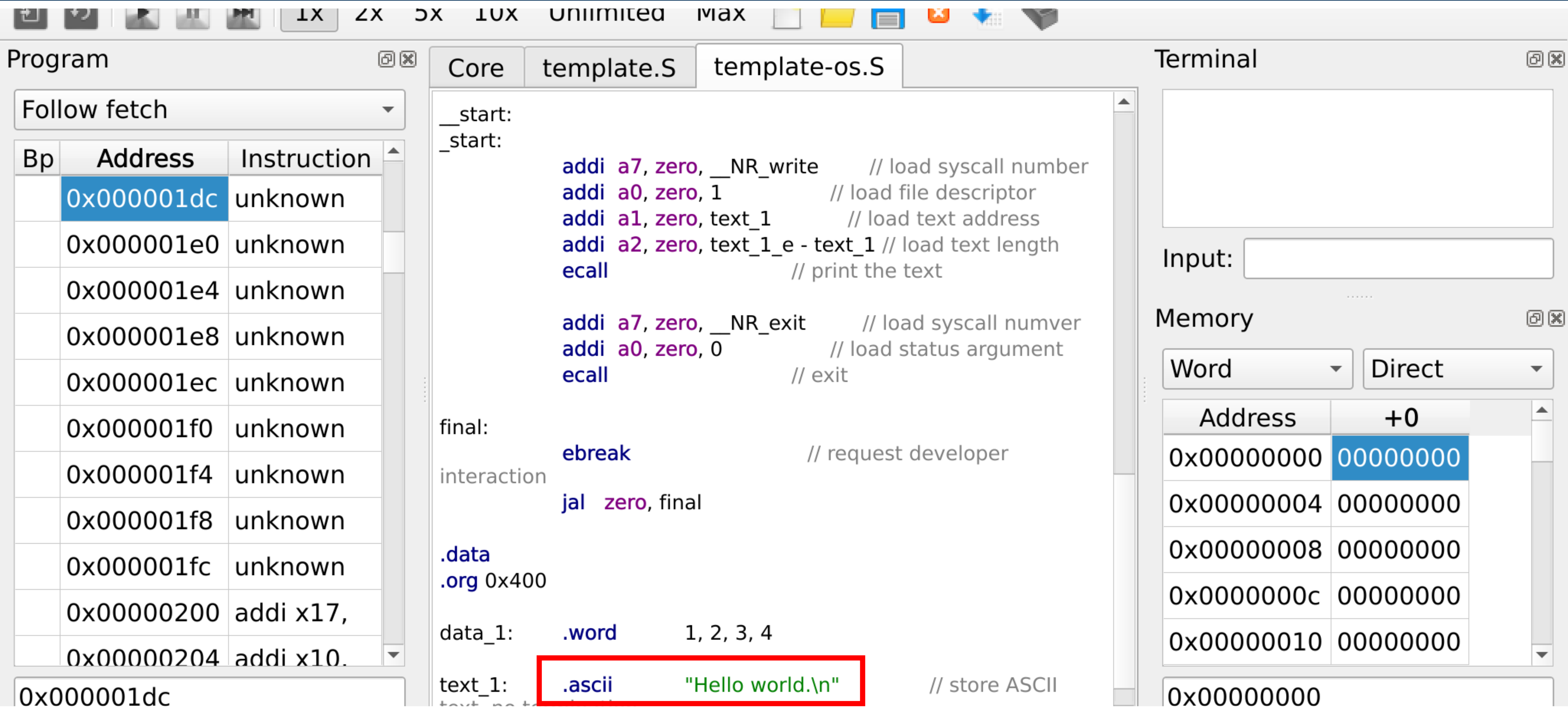
Memory

Word | Direct

| Address | +0 |
|---|---|
| 0x00000000 | 00000000 |
| 0x00000004 | 00000000 |
| 0x00000008 | 00000000 |
| 0x0000000c | 00000000 |
| 0x00000010 | 00000000 |

0x00000000

# OS Template



Program

Follow fetch

| Bp | Address | Instruction |
|---|---|---|
| | 0x000001dc | unknown |
| | 0x000001e0 | unknown |
| | 0x000001e4 | unknown |
| | 0x000001e8 | unknown |
| | 0x000001ec | unknown |
| | 0x000001f0 | unknown |
| | 0x000001f4 | unknown |
| | 0x000001f8 | unknown |
| | 0x000001fc | unknown |
| | 0x00000200 | addi x17, |
| | 0x00000204 | addi x10, |

0x000001dc

Core | template.S | template-os.S

```
__start:
_start:
            addi  a7, zero, __NR_write       // load syscall number
            addi  a0, zero, 1               // load file descriptor
            addi  a1, zero, text_1          // load text address
            addi  a2, zero, text_1_e - text_1 // load text length
            ecall                           // print the text

            addi  a7, zero, __NR_exit       // load syscall numver
            addi  a0, zero, 0               // load status argument
            ecall                           // exit

final:
            ebreak                          // request developer
interaction
            jal   zero, final

.data
.org 0x400

data_1:     .word         1, 2, 3, 4

text_1:     .ascii        "Hello world.\n"          // store ASCII
```

Terminal

Input:

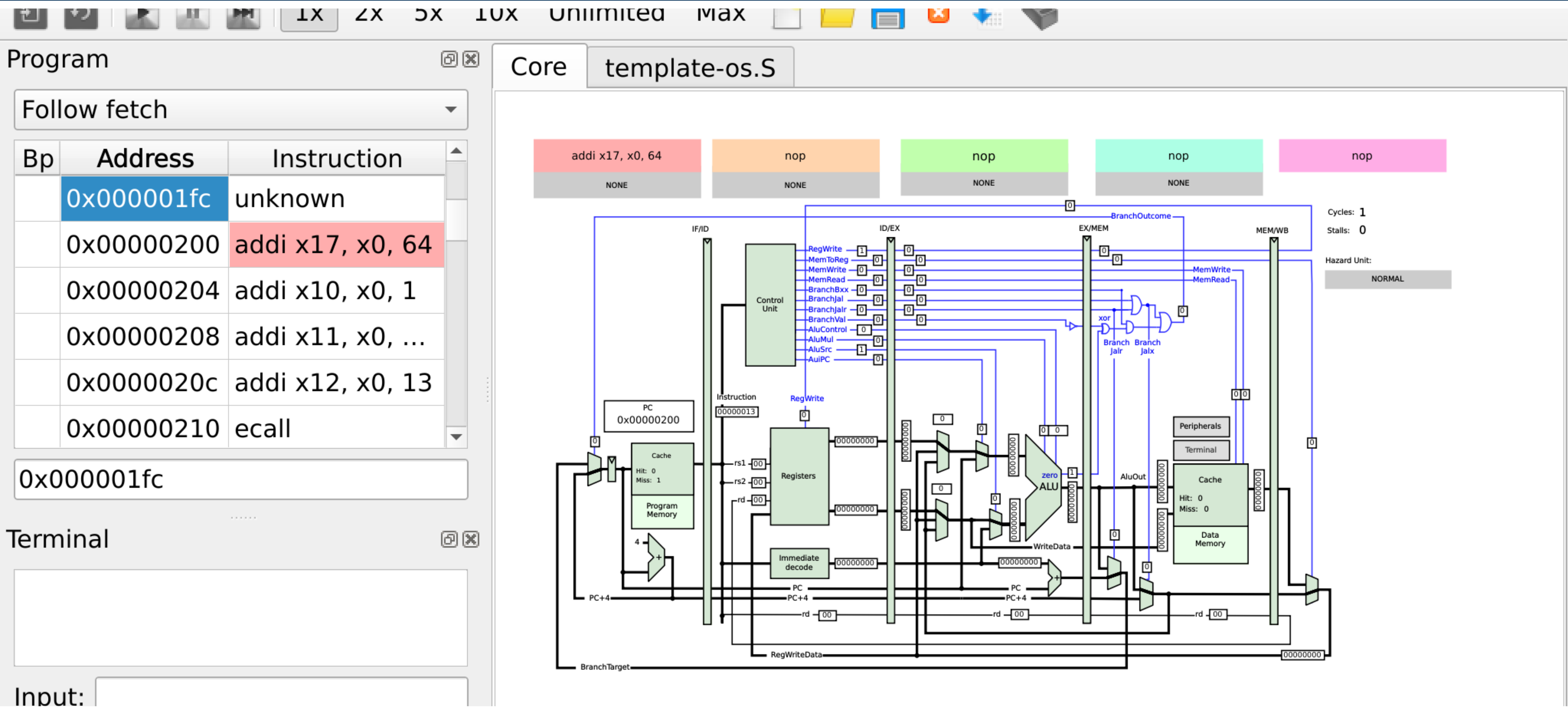
Memory

Word | Direct

| Address | +0 |
|---|---|
| 0x00000000 | 00000000 |
| 0x00000004 | 00000000 |
| 0x00000008 | 00000000 |
| 0x0000000c | 00000000 |
| 0x00000010 | 00000000 |

0x00000000

# Example: Write To File / Terminal



Program

Follow fetch

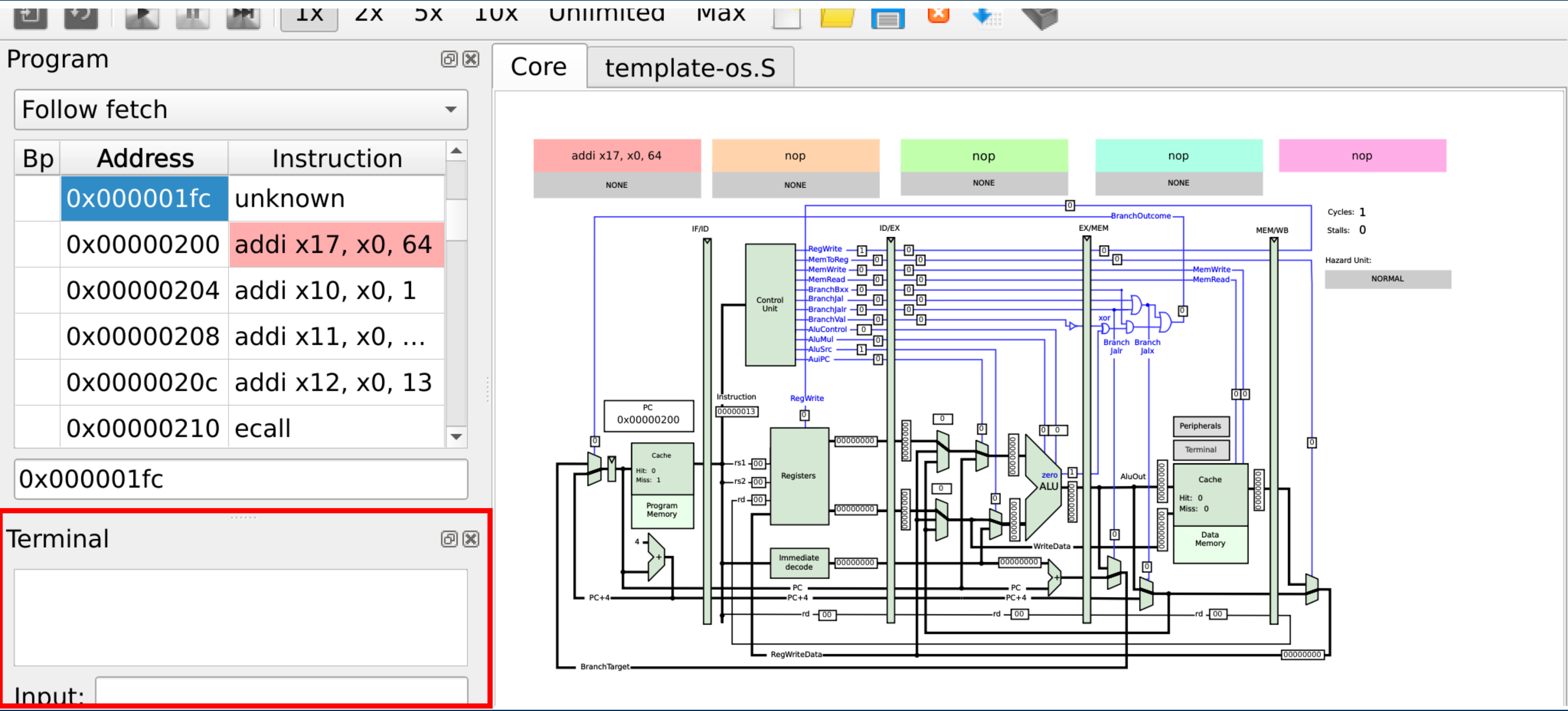
| Bp | Address | Instruction |
|---|---|---|
| | 0x000001fc | unknown |
| | 0x00000200 | addi x17, x0, 64 |
| | 0x00000204 | addi x10, x0, 1 |
| | 0x00000208 | addi x11, x0, … |
| | 0x0000020c | addi x12, x0, 13 |
| | 0x00000210 | ecall |

0x000001fc

Terminal

Input:

# Example: Write To File / Terminal



Program

Follow fetch

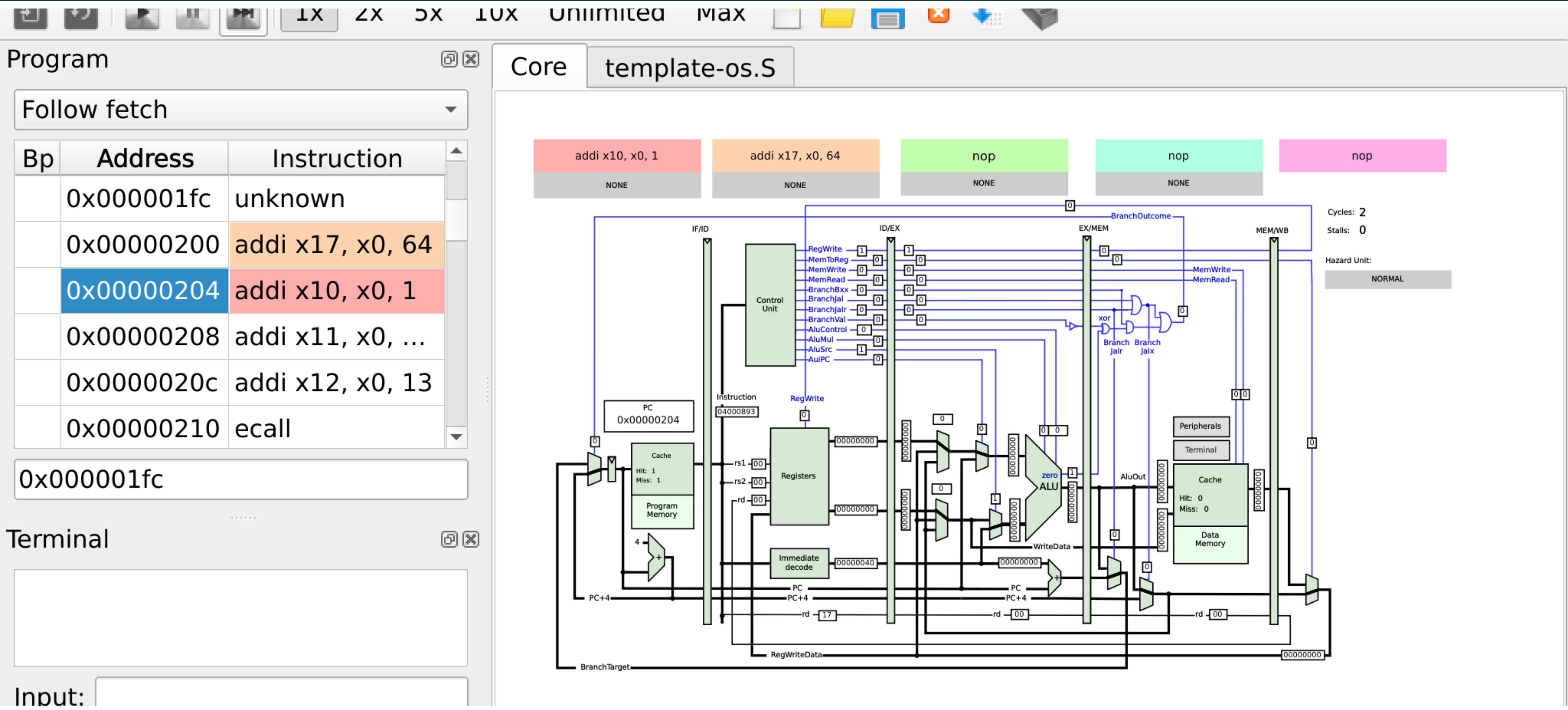
| Bp | Address | Instruction |
|---|---|---|
| | 0x000001fc | unknown |
| | 0x00000200 | addi x17, x0, 64 |
| | 0x00000204 | addi x10, x0, 1 |
| | 0x00000208 | addi x11, x0, ... |
| | 0x0000020c | addi x12, x0, 13 |
| | 0x00000210 | ecall |

0x000001fc

Terminal

Input:

# Example: Write To File / Terminal



Program

Follow fetch

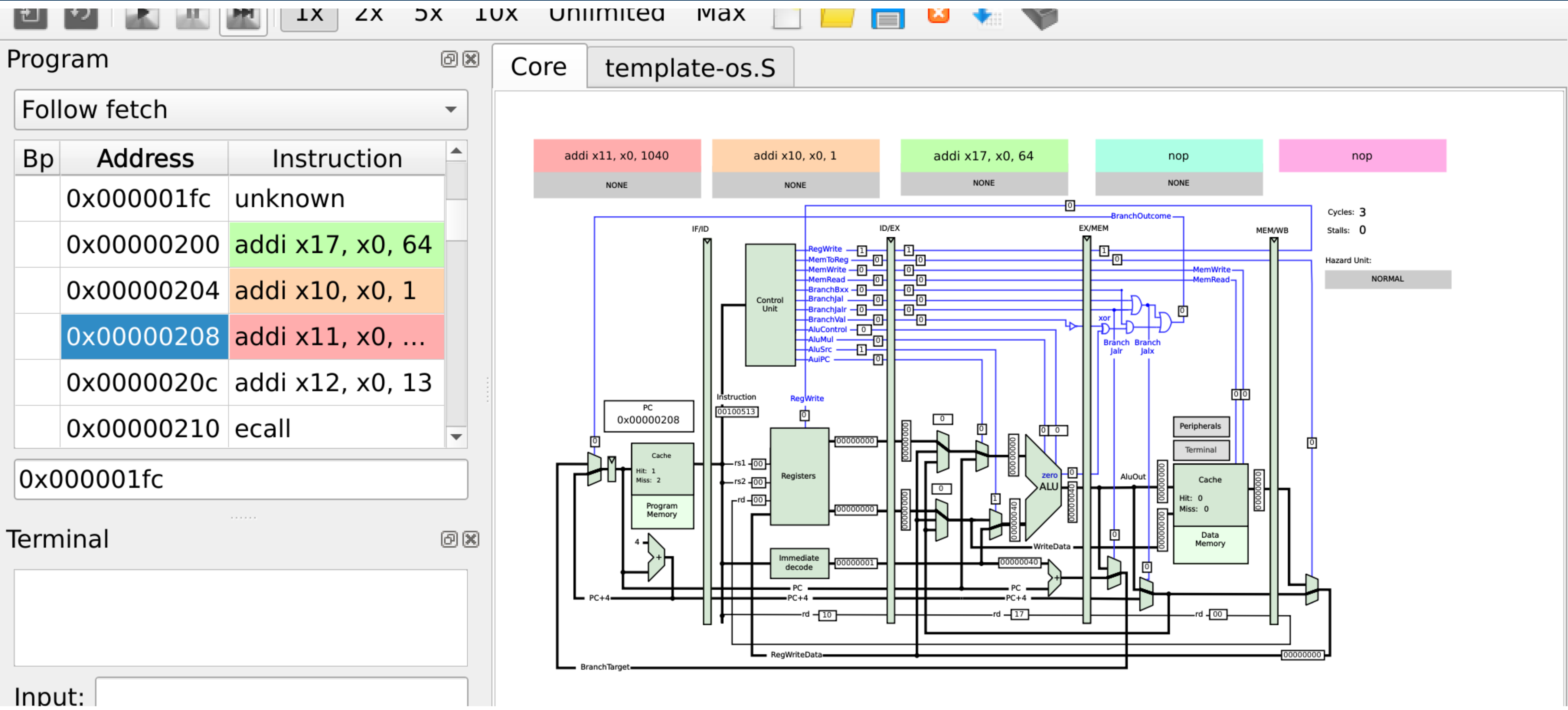
| Bp | Address | Instruction |
|---|---|---|
| | 0x000001fc | unknown |
| | 0x00000200 | addi x17, x0, 64 |
| | 0x00000204 | addi x10, x0, 1 |
| | 0x00000208 | addi x11, x0, ... |
| | 0x0000020c | addi x12, x0, 13 |
| | 0x00000210 | ecall |

0x000001fc

Terminal

Input:

# Example: Write To File / Terminal



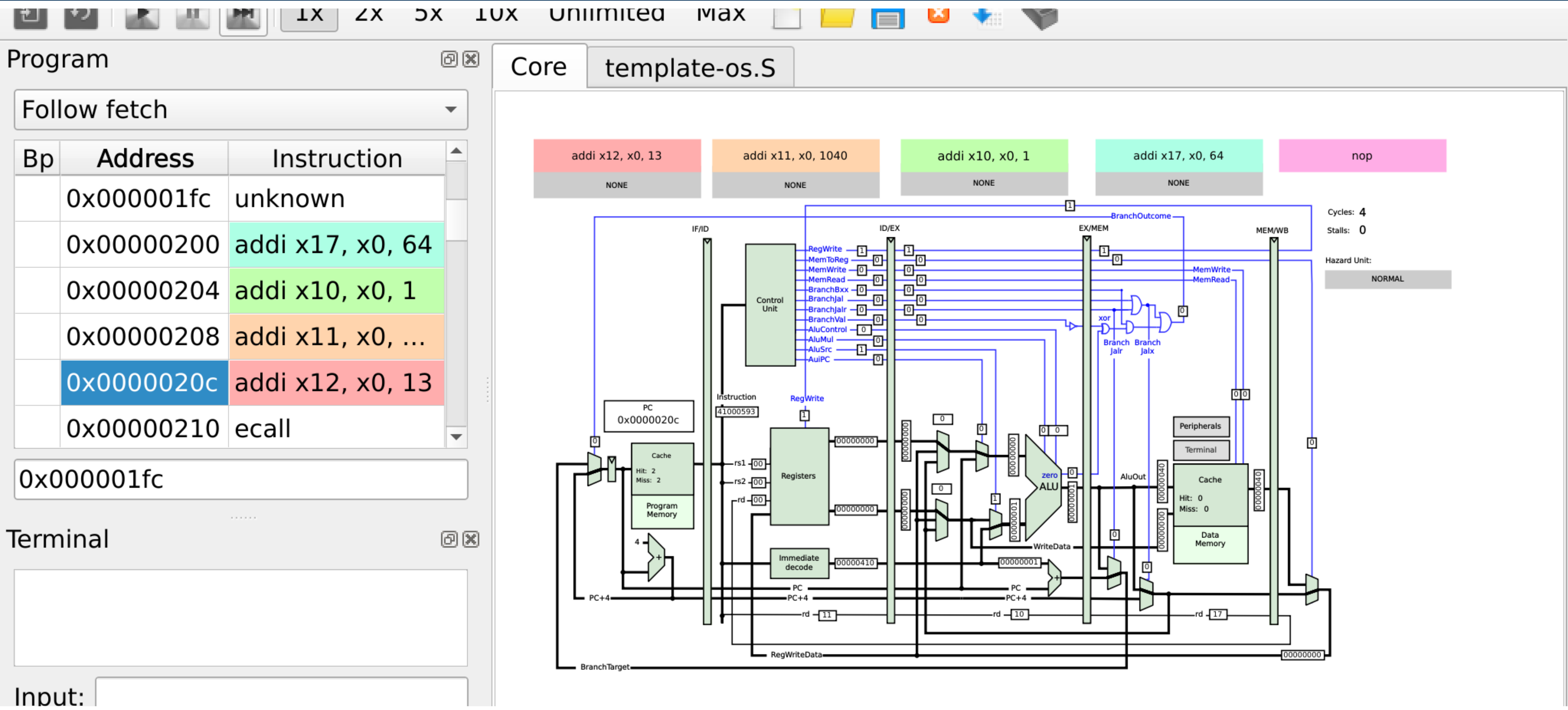
| addi x11, x0, 1040 | addi x10, x0, 1 | addi x17, x0, 64 | nop | nop |
|---|---|---|---|---|
| NONE | NONE | NONE | NONE | |

| Bp | Address | Instruction |
|---|---|---|
| | 0x000001fc | unknown |
| | 0x00000200 | addi x17, x0, 64 |
| | 0x00000204 | addi x10, x0, 1 |
| | 0x00000208 | addi x11, x0, ... |
| | 0x0000020c | addi x12, x0, 13 |
| | 0x00000210 | ecall |

0x000001fc

PC 0x00000208

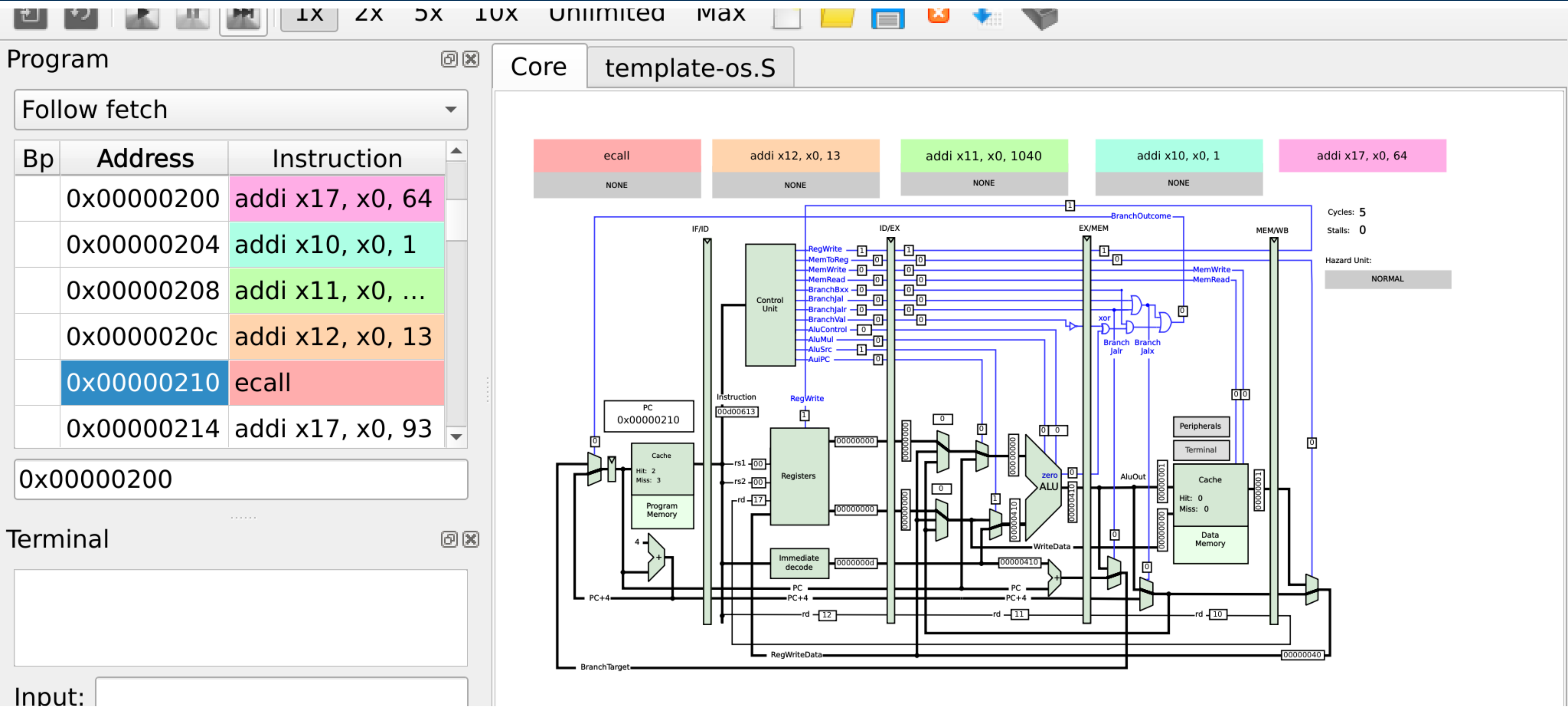
# Example: Write To File / Terminal



| Bp | Address | Instruction |
|---|---|---|
| | 0x000001fc | unknown |
| | 0x00000200 | addi x17, x0, 64 |
| | 0x00000204 | addi x10, x0, 1 |
| | 0x00000208 | addi x11, x0, ... |
| | 0x0000020c | addi x12, x0, 13 |
| | 0x00000210 | ecall |

0x000001fc

# Example: Write To File / Terminal



Program

Follow fetch

| Bp | Address | Instruction |
|---|---|---|
| | 0x00000200 | addi x17, x0, 64 |
| | 0x00000204 | addi x10, x0, 1 |
| | 0x00000208 | addi x11, x0, ... |
| | 0x0000020c | addi x12, x0, 13 |
| | 0x00000210 | ecall |
| | 0x00000214 | addi x17, x0, 93 |

0x00000200

Terminal

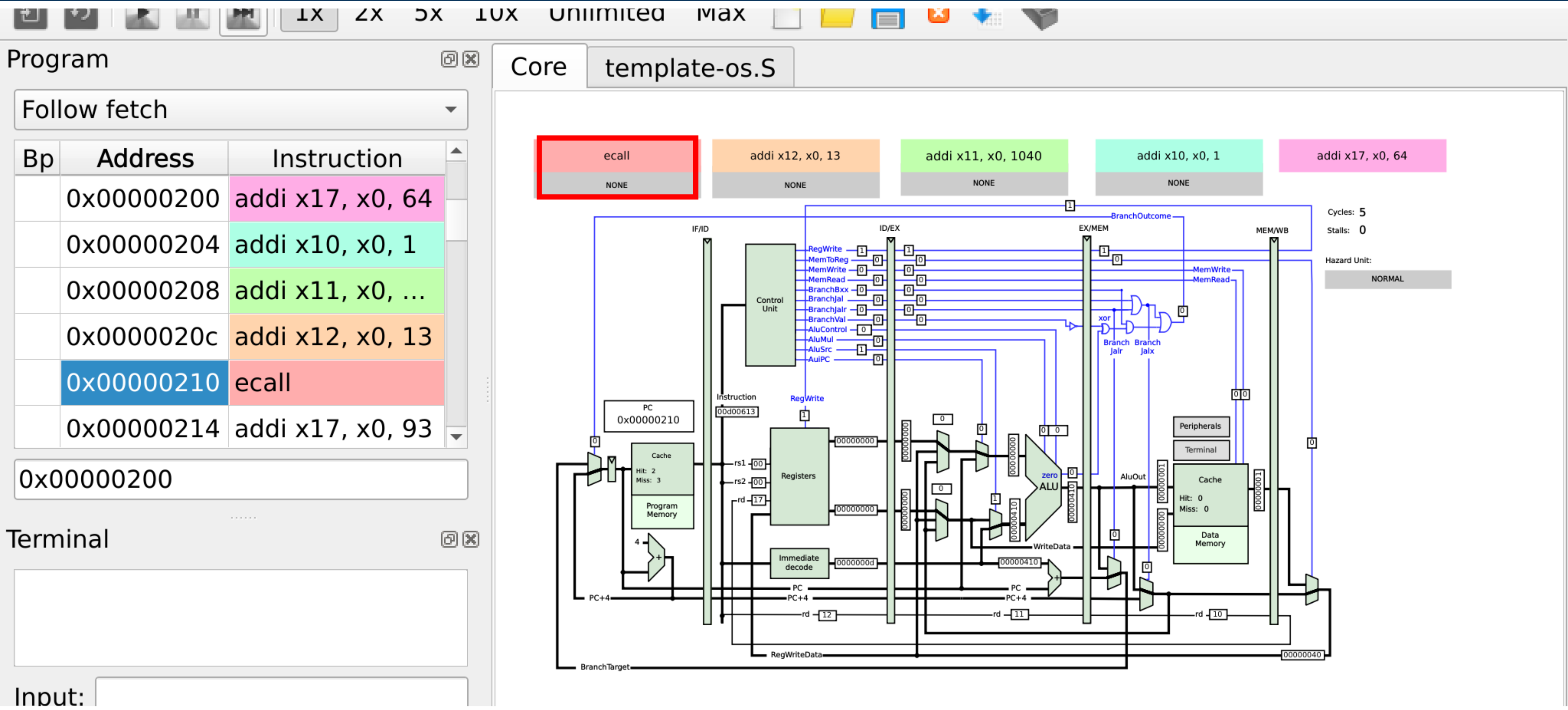
Input:

Core | template-os.S

| ecall | addi x12, x0, 13 | addi x11, x0, 1040 | addi x10, x0, 1 | addi x17, x0, 64 |
|---|---|---|---|---|
| NONE | NONE | NONE | NONE | |

Cycles: 5
Stalls: 0
Hazard Unit: NORMAL

# Example: Write To File / Terminal



Program

Follow fetch

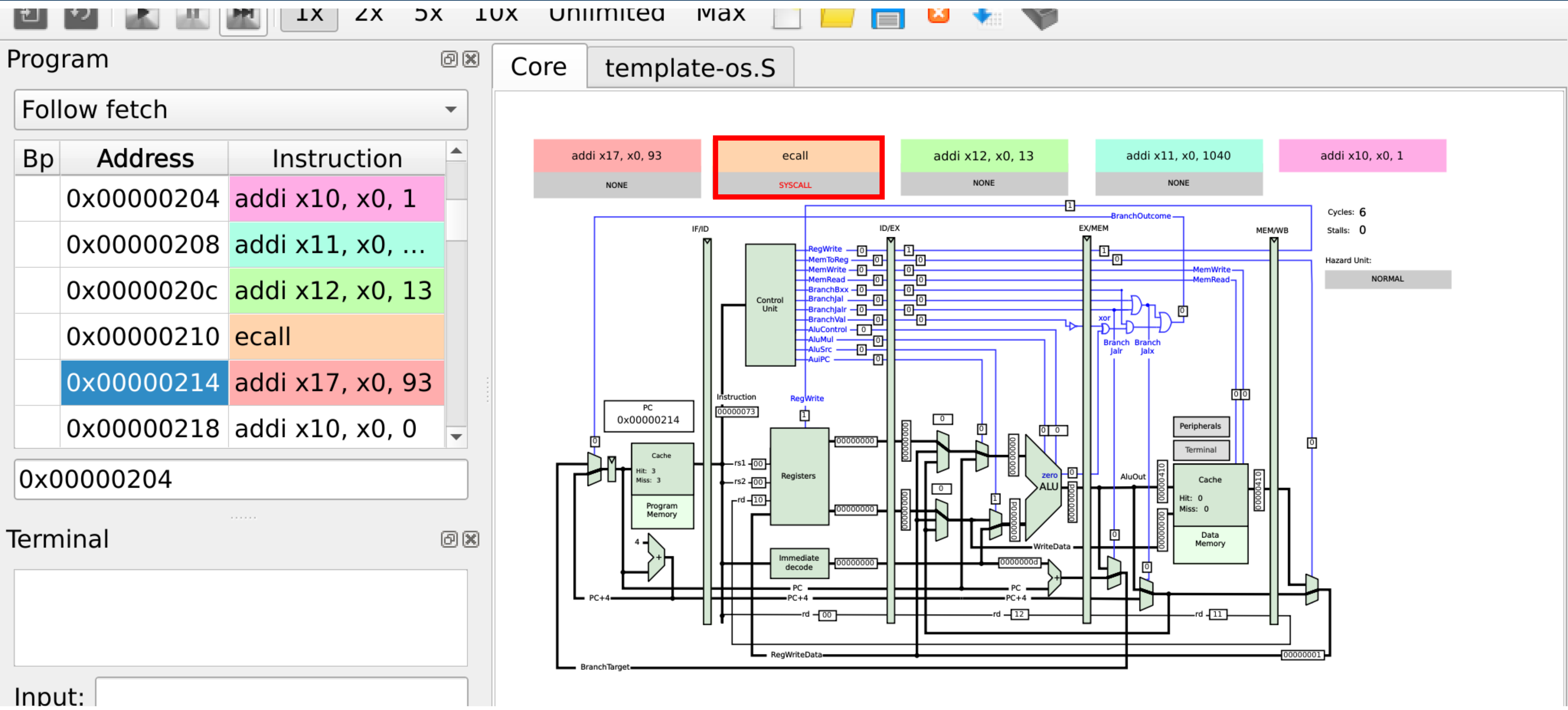
| Bp | Address | Instruction |
|---|---|---|
| | 0x00000200 | addi x17, x0, 64 |
| | 0x00000204 | addi x10, x0, 1 |
| | 0x00000208 | addi x11, x0, ... |
| | 0x0000020c | addi x12, x0, 13 |
| | 0x00000210 | ecall |
| | 0x00000214 | addi x17, x0, 93 |

0x00000200

Terminal

Input:

# Example: Write To File / Terminal



| Bp | Address | Instruction |
|---|---|---|
| | 0x00000204 | addi x10, x0, 1 |
| | 0x00000208 | addi x11, x0, ... |
| | 0x0000020c | addi x12, x0, 13 |
| | 0x00000210 | ecall |
| | 0x00000214 | addi x17, x0, 93 |
| | 0x00000218 | addi x10, x0, 0 |

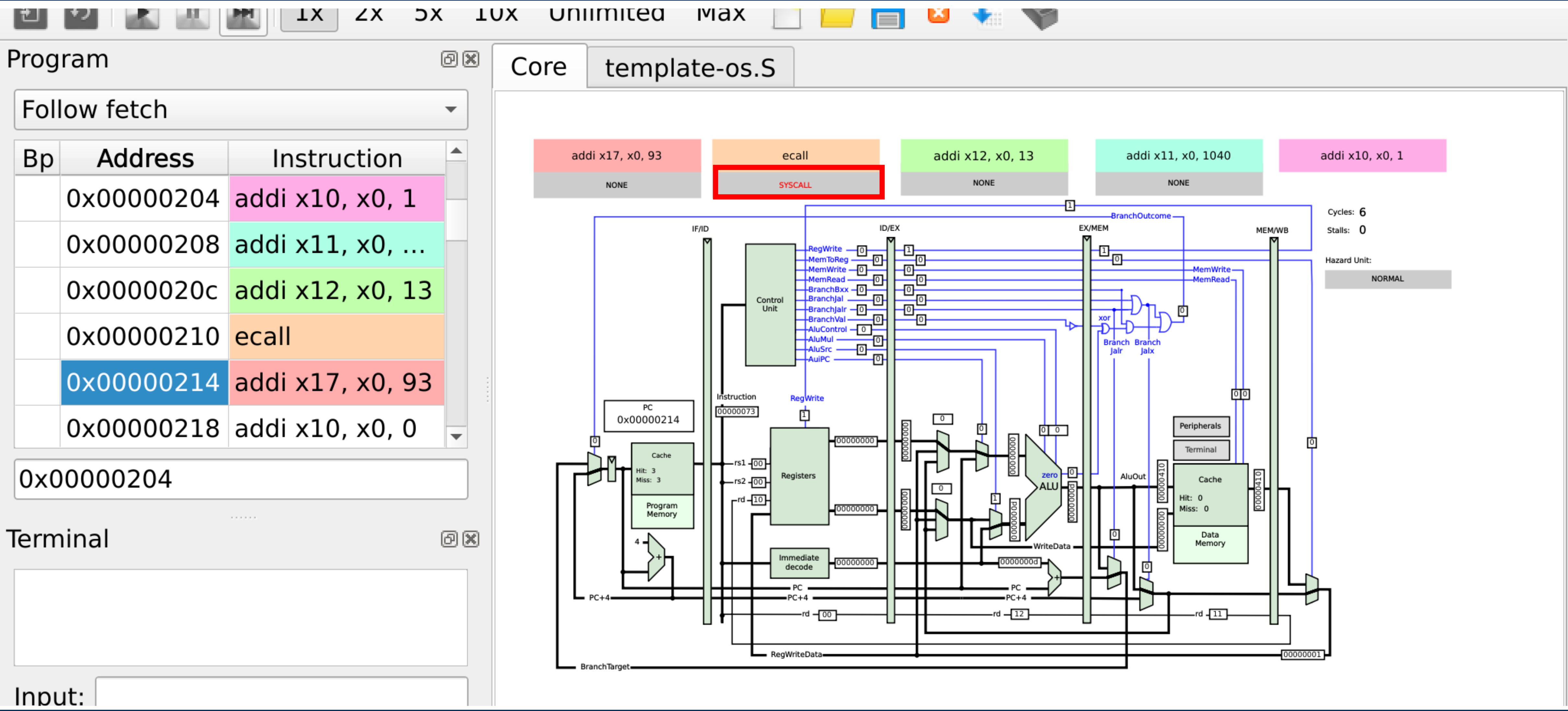
0x00000204

| addi x17, x0, 93 | ecall | addi x12, x0, 13 | addi x11, x0, 1040 | addi x10, x0, 1 |
|---|---|---|---|---|
| NONE | SYSCALL | NONE | NONE | |

Cycles: 6
Stalls: 0
Hazard Unit: NORMAL

PC 0x00000214

# Example: Write To File / Terminal



Program

Follow fetch

| Bp | Address | Instruction |
|---|---|---|
| | 0x00000204 | addi x10, x0, 1 |
| | 0x00000208 | addi x11, x0, ... |
| | 0x0000020c | addi x12, x0, 13 |
| | 0x00000210 | ecall |
| | 0x00000214 | addi x17, x0, 93 |
| | 0x00000218 | addi x10, x0, 0 |

0x00000204

Terminal

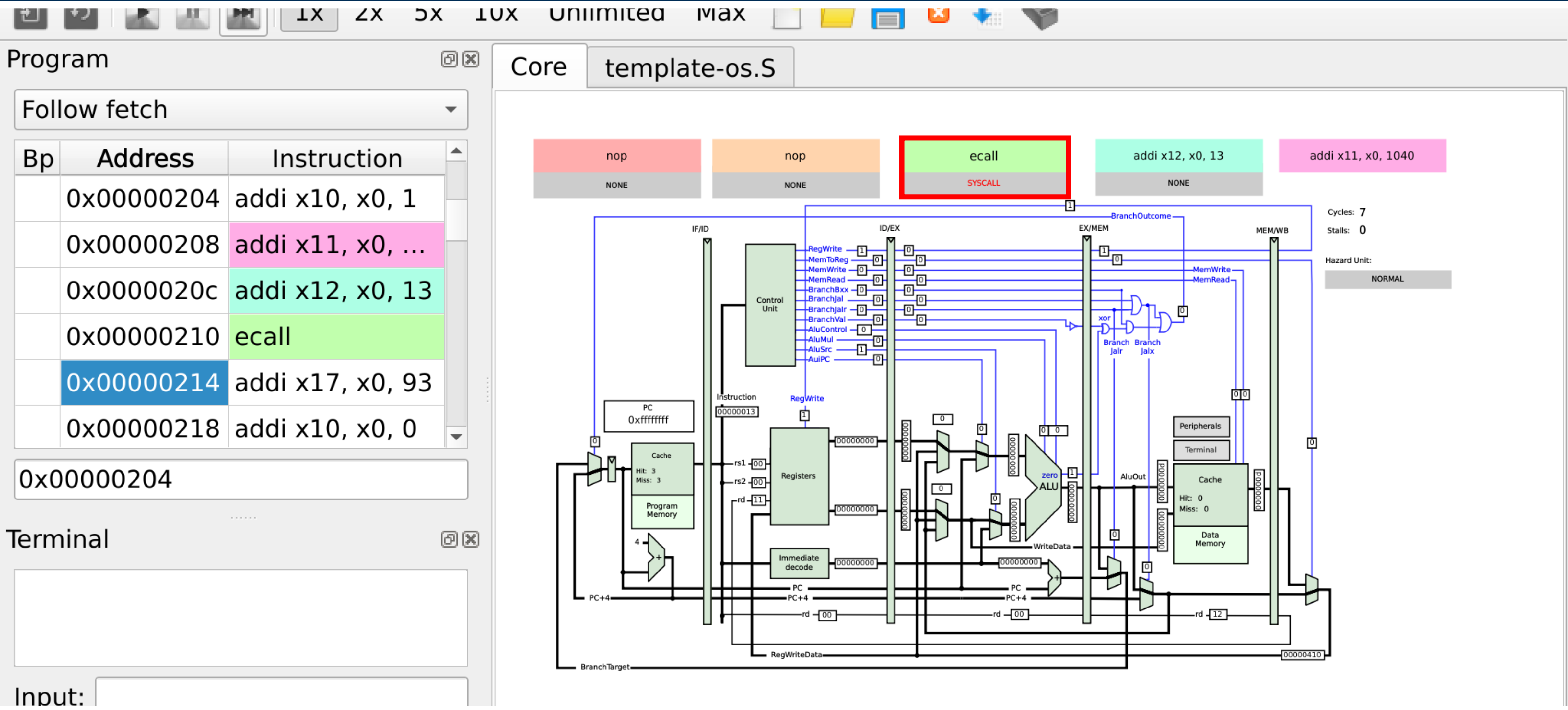
Input:

Core | template-os.S

| addi x17, x0, 93 | ecall | addi x12, x0, 13 | addi x11, x0, 1040 | addi x10, x0, 1 |
|---|---|---|---|---|
| NONE | SYSCALL | NONE | NONE | |

Cycles: 6
Stalls: 0
Hazard Unit: NORMAL

# Example: Write To File / Terminal



Program

Follow fetch

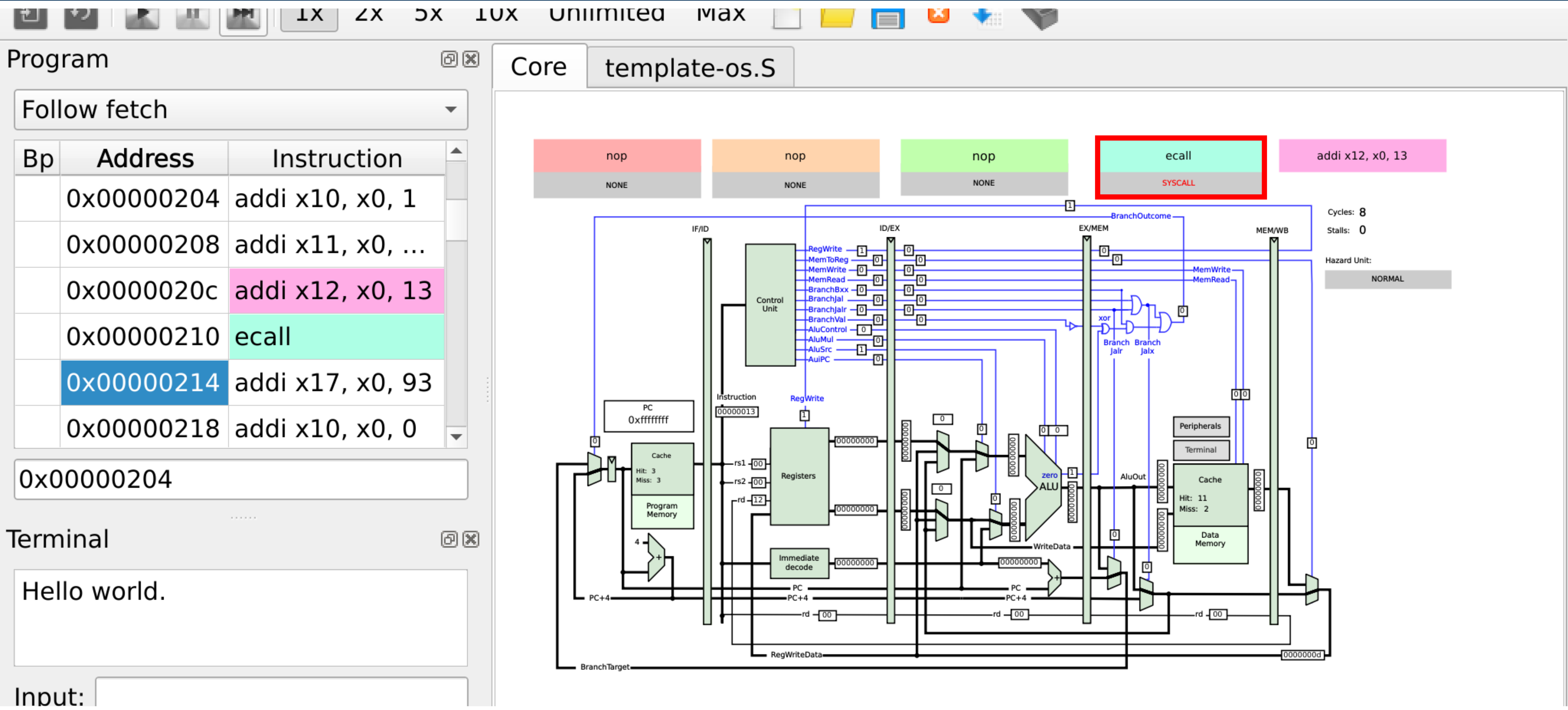
| Bp | Address | Instruction |
|---|---|---|
| | 0x00000204 | addi x10, x0, 1 |
| | 0x00000208 | addi x11, x0, ... |
| | 0x0000020c | addi x12, x0, 13 |
| | 0x00000210 | ecall |
| | 0x00000214 | addi x17, x0, 93 |
| | 0x00000218 | addi x10, x0, 0 |

0x00000204

Terminal

Input:

# Example: Write To File / Terminal



Program

Follow fetch

| Bp | Address | Instruction |
|---|---|---|
| | 0x00000204 | addi x10, x0, 1 |
| | 0x00000208 | addi x11, x0, ... |
| | 0x0000020c | addi x12, x0, 13 |
| | 0x00000210 | ecall |
| | 0x00000214 | addi x17, x0, 93 |
| | 0x00000218 | addi x10, x0, 0 |

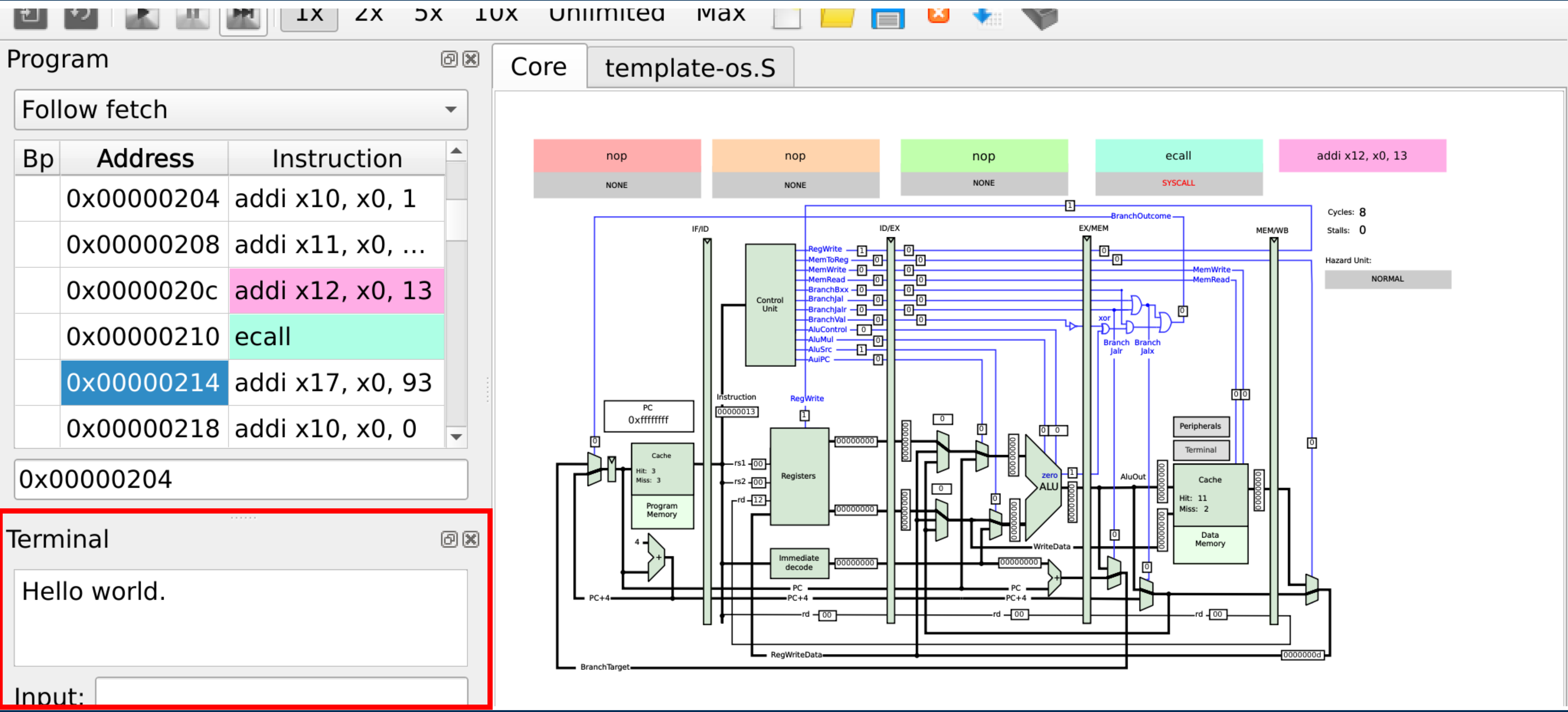
0x00000204

Terminal

```
Hello world.
```

Input:

# Example: Write To File / Terminal



Program

Follow fetch

| Bp | Address | Instruction |
|---|---|---|
| | 0x00000204 | addi x10, x0, 1 |
| | 0x00000208 | addi x11, x0, ... |
| | 0x0000020c | addi x12, x0, 13 |
| | 0x00000210 | ecall |
| | 0x00000214 | addi x17, x0, 93 |
| | 0x00000218 | addi x10, x0, 0 |

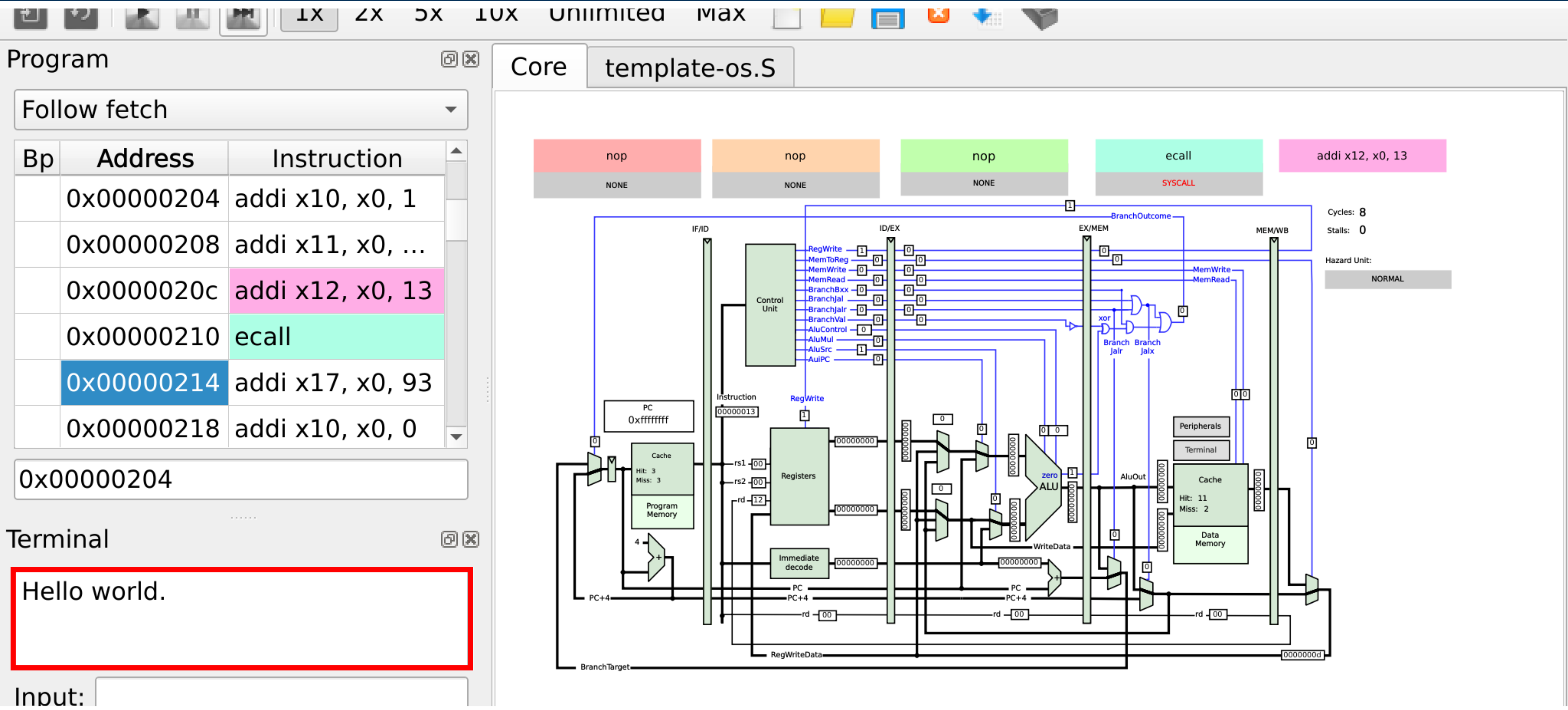
0x00000204

Terminal

```
Hello world.
```

Input:

# Example: Write To File / Terminal

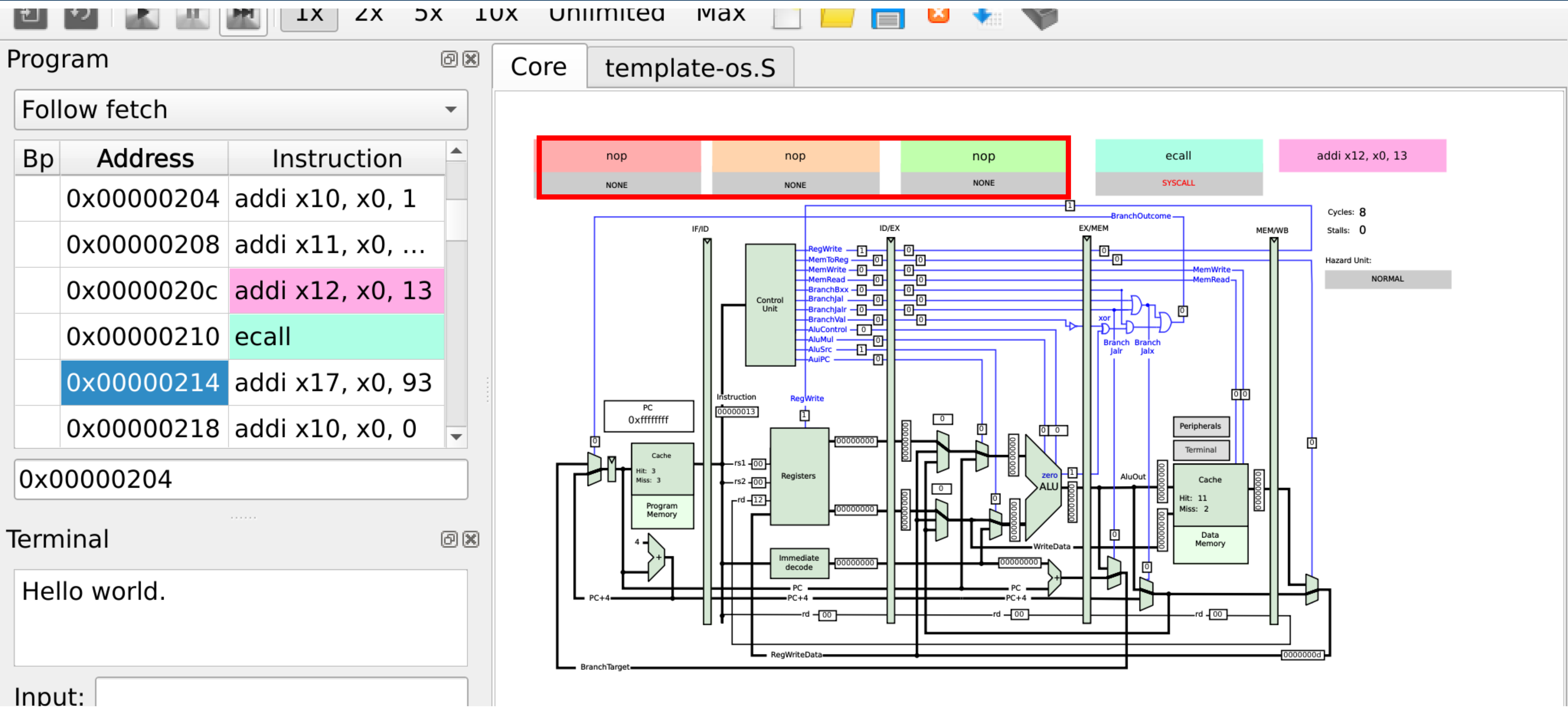


| Bp | Address | Instruction |
|---|---|---|
| | 0x00000204 | addi x10, x0, 1 |
| | 0x00000208 | addi x11, x0, ... |
| | 0x0000020c | addi x12, x0, 13 |
| | 0x00000210 | ecall |
| | 0x00000214 | addi x17, x0, 93 |
| | 0x00000218 | addi x10, x0, 0 |

0x00000204

```
Hello world.
```

# Example: Write To File / Terminal



Program

Follow fetch

| Bp | Address | Instruction |
|---|---|---|
| | 0x00000204 | addi x10, x0, 1 |
| | 0x00000208 | addi x11, x0, ... |
| | 0x0000020c | addi x12, x0, 13 |
| | 0x00000210 | ecall |
| | 0x00000214 | addi x17, x0, 93 |
| | 0x00000218 | addi x10, x0, 0 |

0x00000204

Terminal

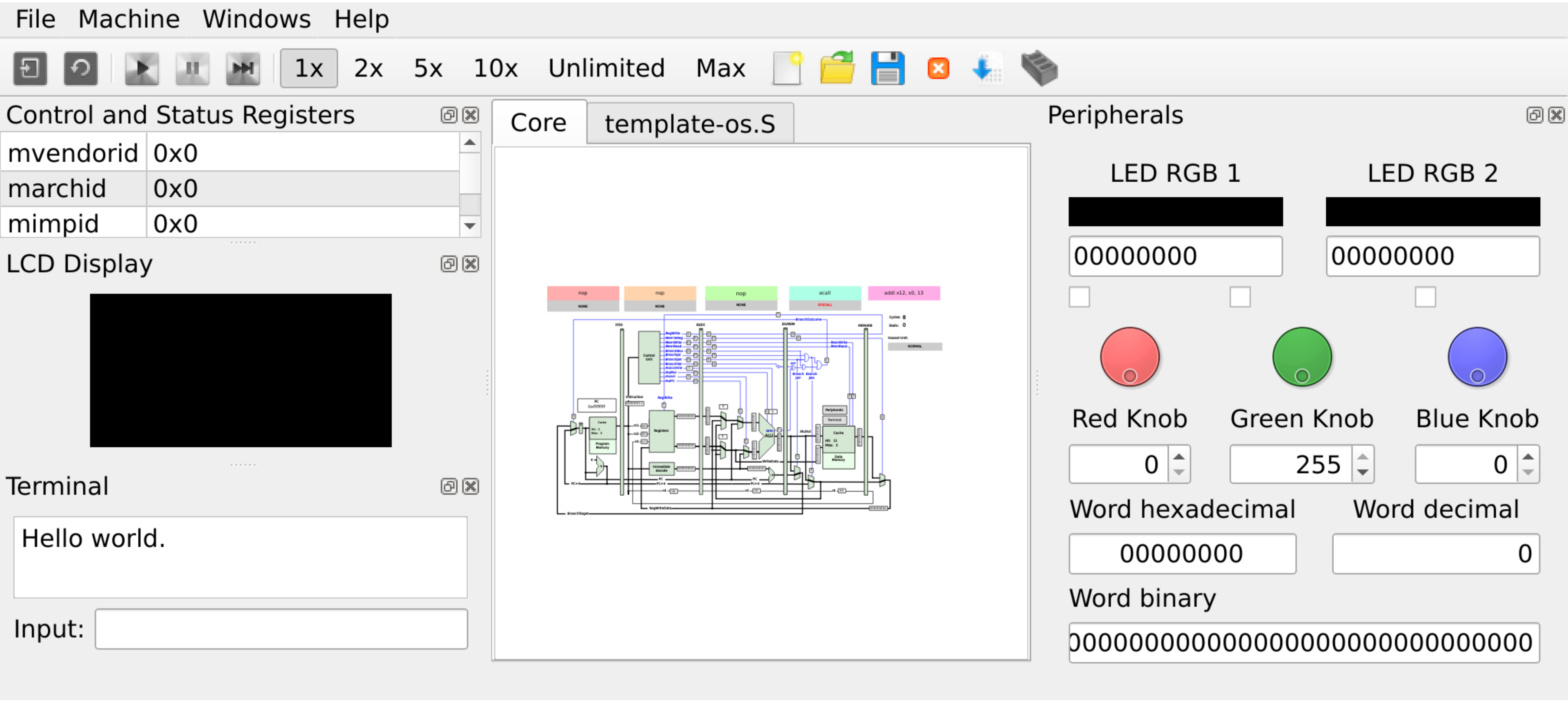
```
Hello world.
```

Input:

# Supported Syscalls

```
void exit(int status)
ssize_t read(int fd, void *buf, size_t count) __N
ssize_t write(int fd, const void *buf, size_t count)
int close(int fd)
int openat(int dirfd, const char *pathname, int flags, mode_t
 mode)
void * brk(void *addr)
int ftruncate(int fd, off_t length)
ssize_t readv(int fd, const struct iovec *iov, int iovcnt)
ssize_t writev(int fd, const struct iovec *iov, int iovcnt)
```

# Peripherals and CSR



Control and Status Registers

| | |
|---|---|
| mvendorid | 0x0 |
| marchid | 0x0 |
| mimpid | 0x0 |

LCD Display

Terminal

```
Hello world.
```

Input:

Peripherals

| LED RGB 1 | LED RGB 2 |
|---|---|
| 00000000 | 00000000 |

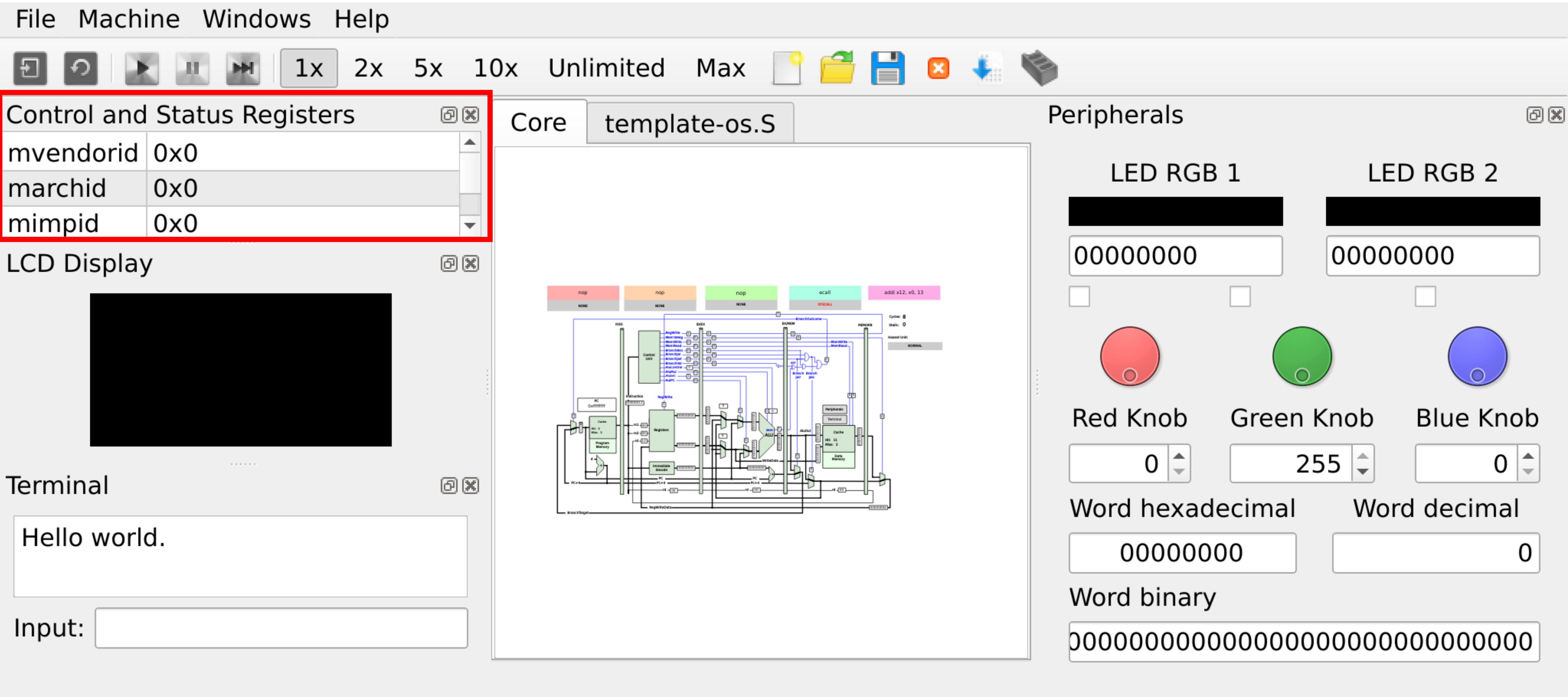
| Red Knob | Green Knob | Blue Knob |
|---|---|---|
| 0 | 255 | 0 |

Word hexadecimal: 00000000

Word decimal: 0

Word binary: 00000000000000000000000000000000

# Peripherals and CSR



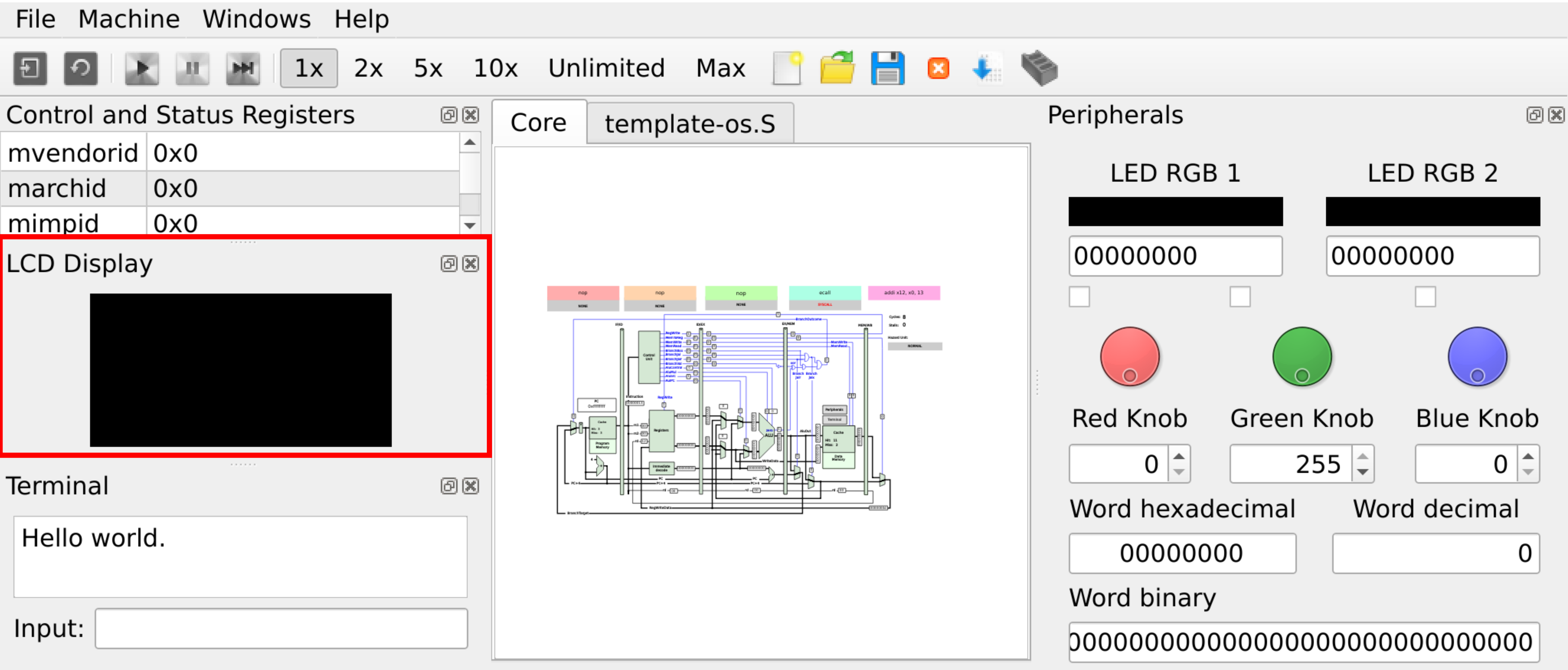
Control and Status Registers

| | |
|---|---|
| mvendorid | 0x0 |
| marchid | 0x0 |
| mimpid | 0x0 |

Hello world.

LED RGB 1: 00000000
LED RGB 2: 00000000
Red Knob: 0
Green Knob: 255
Blue Knob: 0
Word hexadecimal: 00000000
Word decimal: 0
Word binary: 00000000000000000000000000000000

# Peripherals and CSR



| | |
|---|---|
| mvendorid | 0x0 |
| marchid | 0x0 |
| mimpid | 0x0 |

```
Hello world.
```

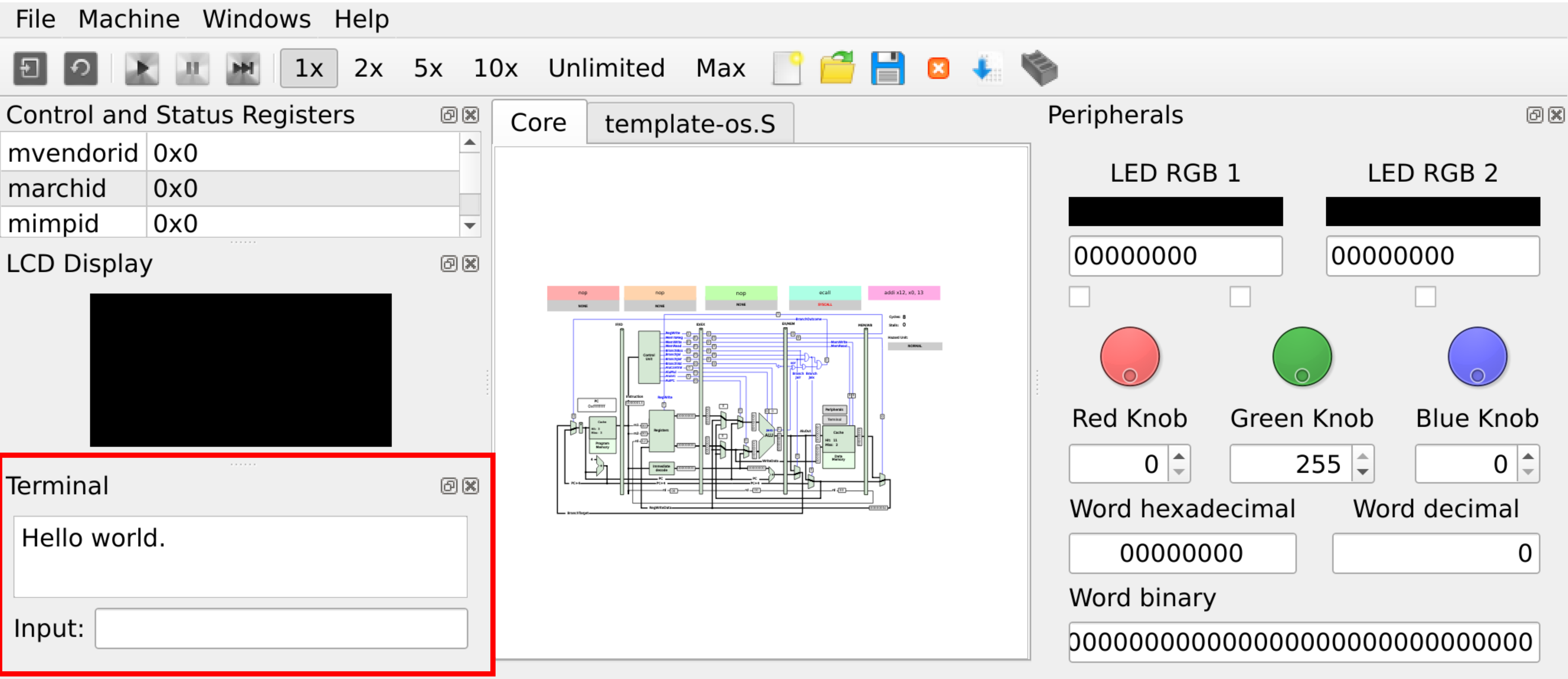
Input:

LED RGB 1: 00000000
LED RGB 2: 00000000

Red Knob: 0
Green Knob: 255
Blue Knob: 0

Word hexadecimal: 00000000
Word decimal: 0
Word binary: 00000000000000000000000000000000

# Peripherals and CSR



Control and Status Registers

| | |
|---|---|
| mvendorid | 0x0 |
| marchid | 0x0 |
| mimpid | 0x0 |

Terminal

```
Hello world.
```

Input:

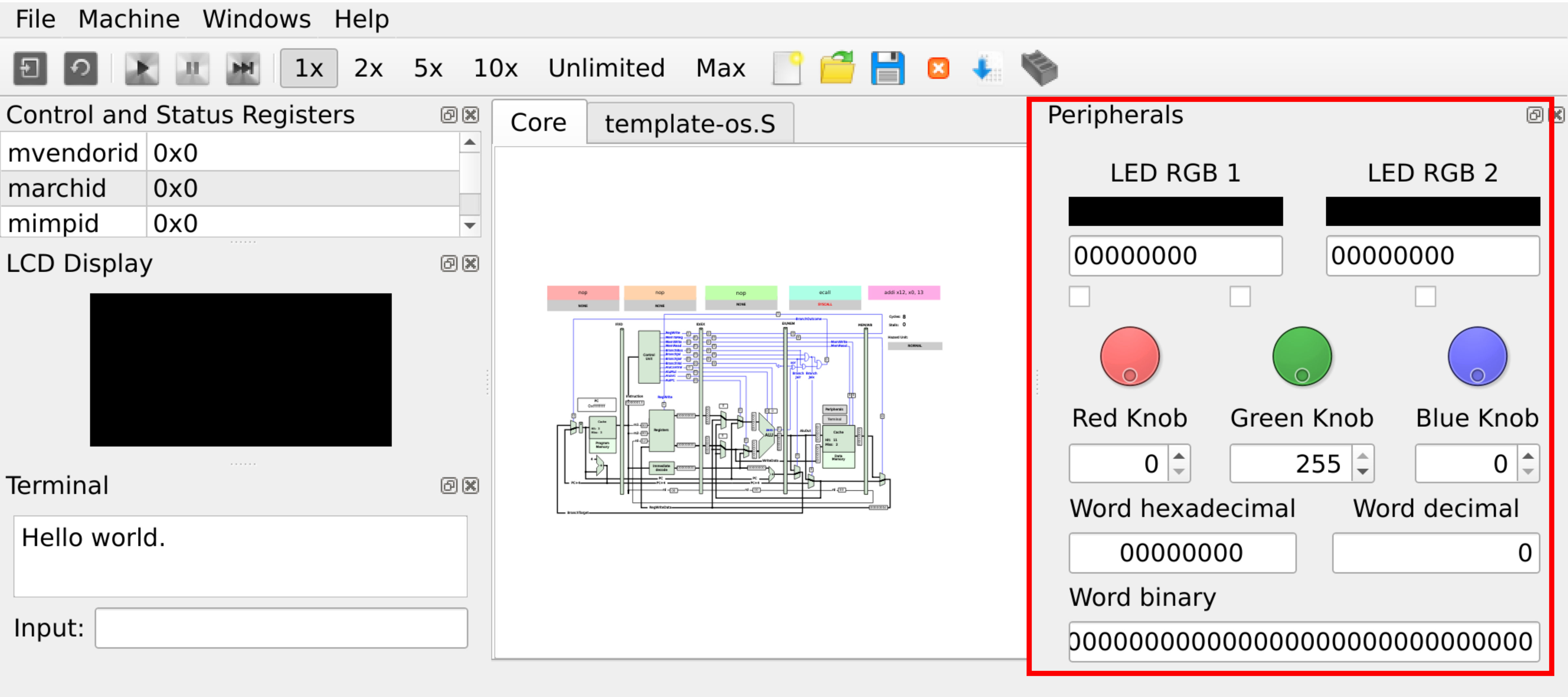
Peripherals

LED RGB 1: 00000000
LED RGB 2: 00000000

Red Knob: 0
Green Knob: 255
Blue Knob: 0

Word hexadecimal: 00000000
Word decimal: 0
Word binary: 00000000000000000000000000000000

# Peripherals and CSR



Control and Status Registers

| | |
|---|---|
| mvendorid | 0x0 |
| marchid | 0x0 |
| mimpid | 0x0 |

LCD Display

Terminal

```
Hello world.
```

Input:

Peripherals

| LED RGB 1 | LED RGB 2 |
|---|---|
| 00000000 | 00000000 |

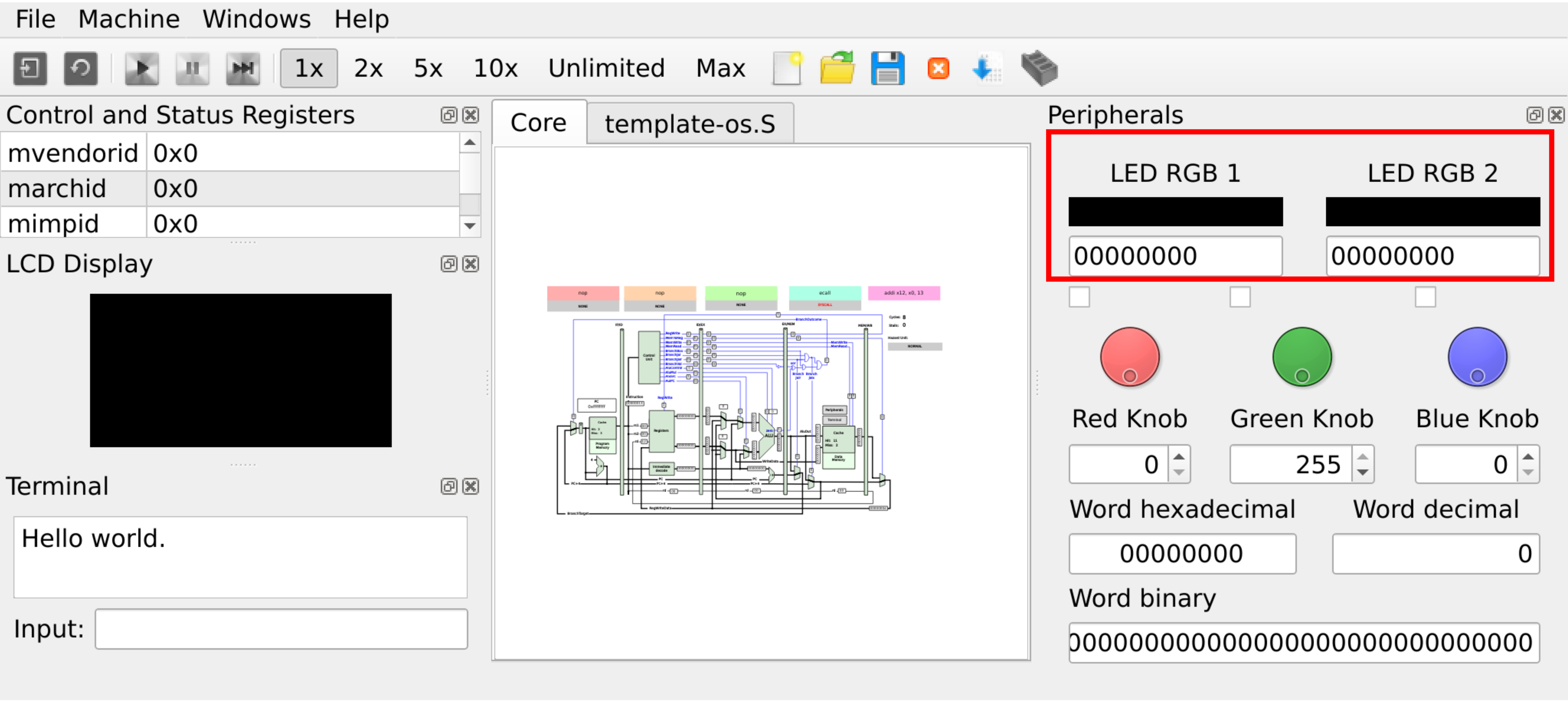
| Red Knob | Green Knob | Blue Knob |
|---|---|---|
| 0 | 255 | 0 |

Word hexadecimal: 00000000

Word decimal: 0

Word binary: 00000000000000000000000000000000

# Peripherals and CSR



Control and Status Registers

| | |
|---|---|
| mvendorid | 0x0 |
| marchid | 0x0 |
| mimpid | 0x0 |

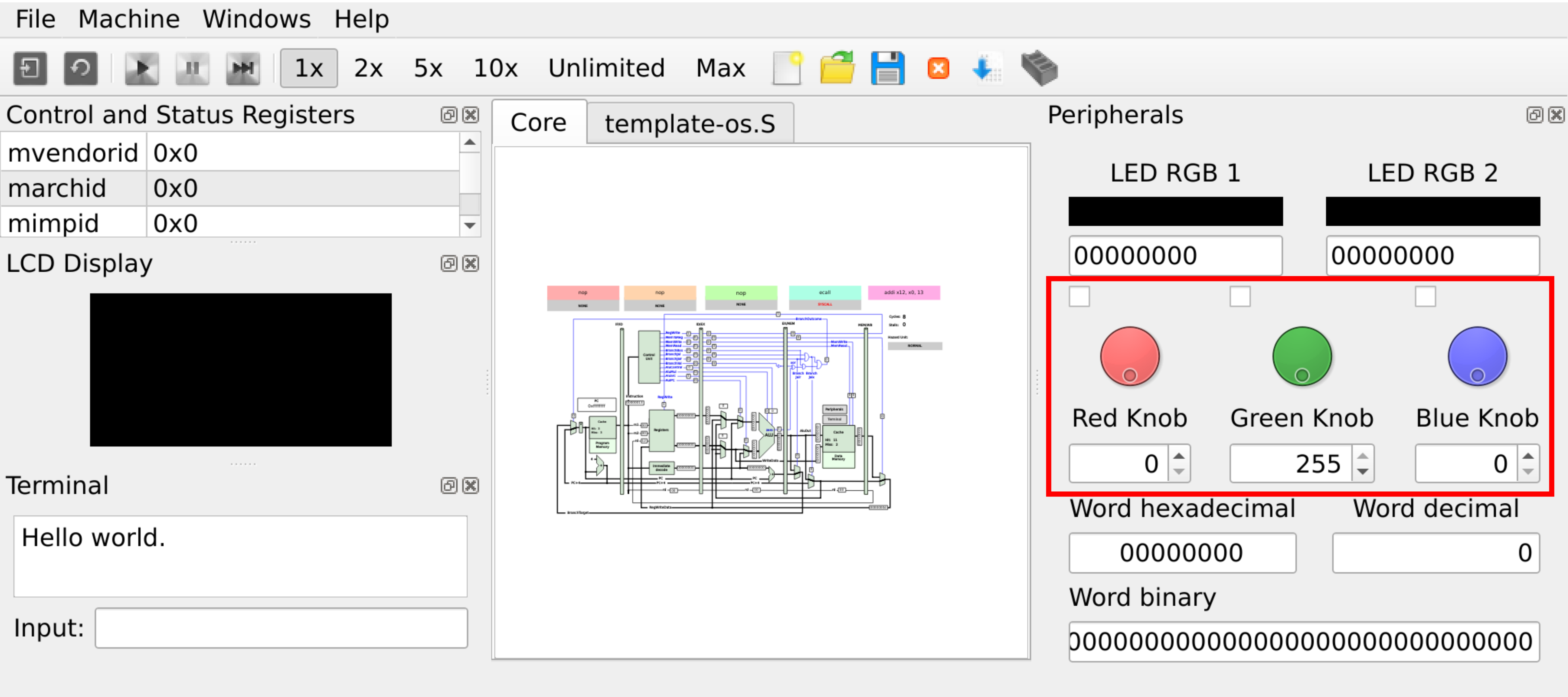
```
Hello world.
```

LED RGB 1: 00000000
LED RGB 2: 00000000

Red Knob: 0
Green Knob: 255
Blue Knob: 0

Word hexadecimal: 00000000
Word decimal: 0
Word binary: 00000000000000000000000000000000

# Peripherals and CSR



File Machine Windows Help

1x 2x 5x 10x Unlimited Max

Control and Status Registers

| | |
|---|---|
| mvendorid | 0x0 |
| marchid | 0x0 |
| mimpid | 0x0 |

LCD Display

Terminal

```
Hello world.
```

Input:

Core | template-os.S

Peripherals

| LED RGB 1 | LED RGB 2 |
|---|---|
| 00000000 | 00000000 |

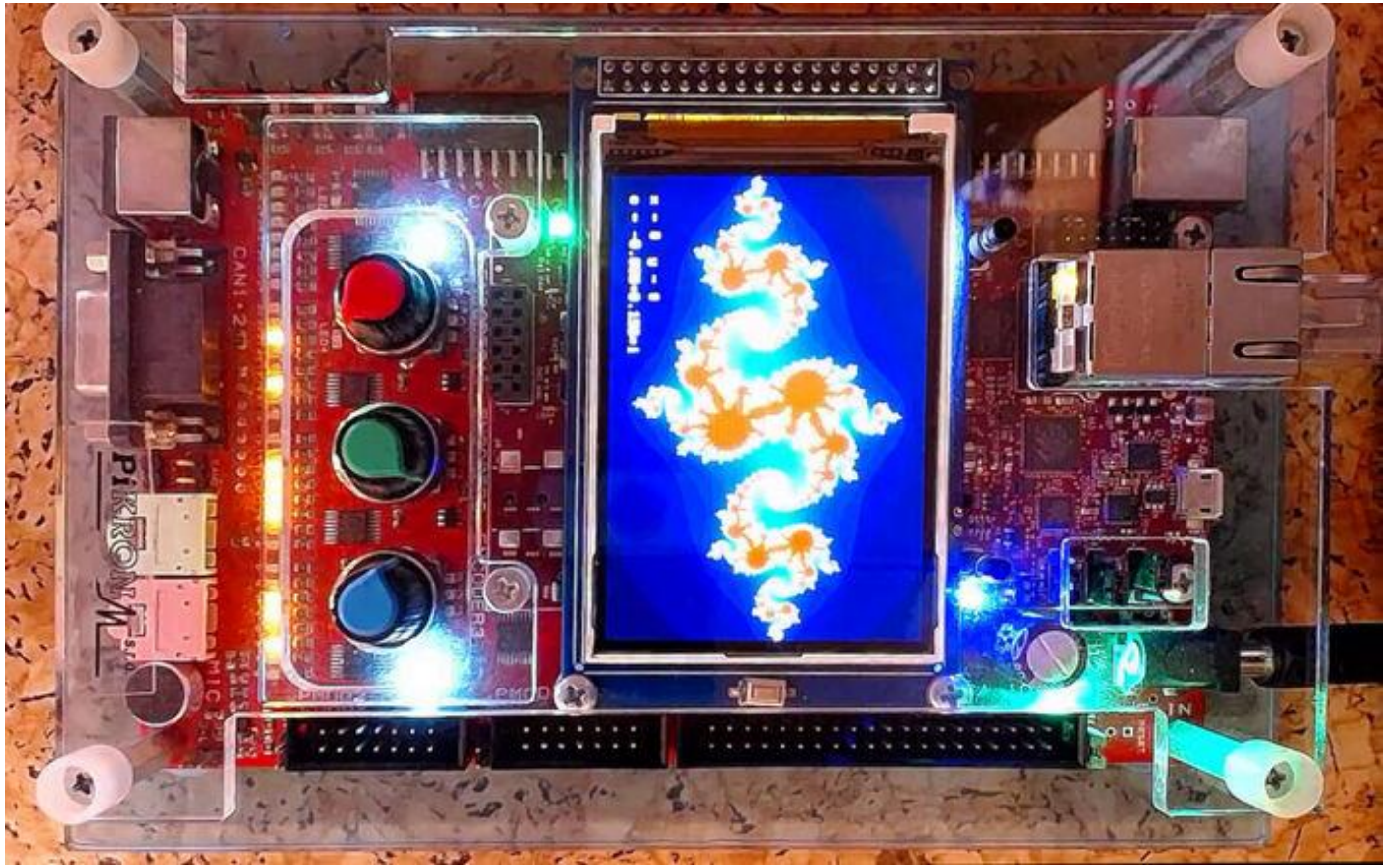
| Red Knob | Green Knob | Blue Knob |
|---|---|---|
| 0 | 255 | 0 |

Word hexadecimal: 00000000

Word decimal: 0

Word binary: 00000000000000000000000000000000

# Motivation for Peripherals Emulation

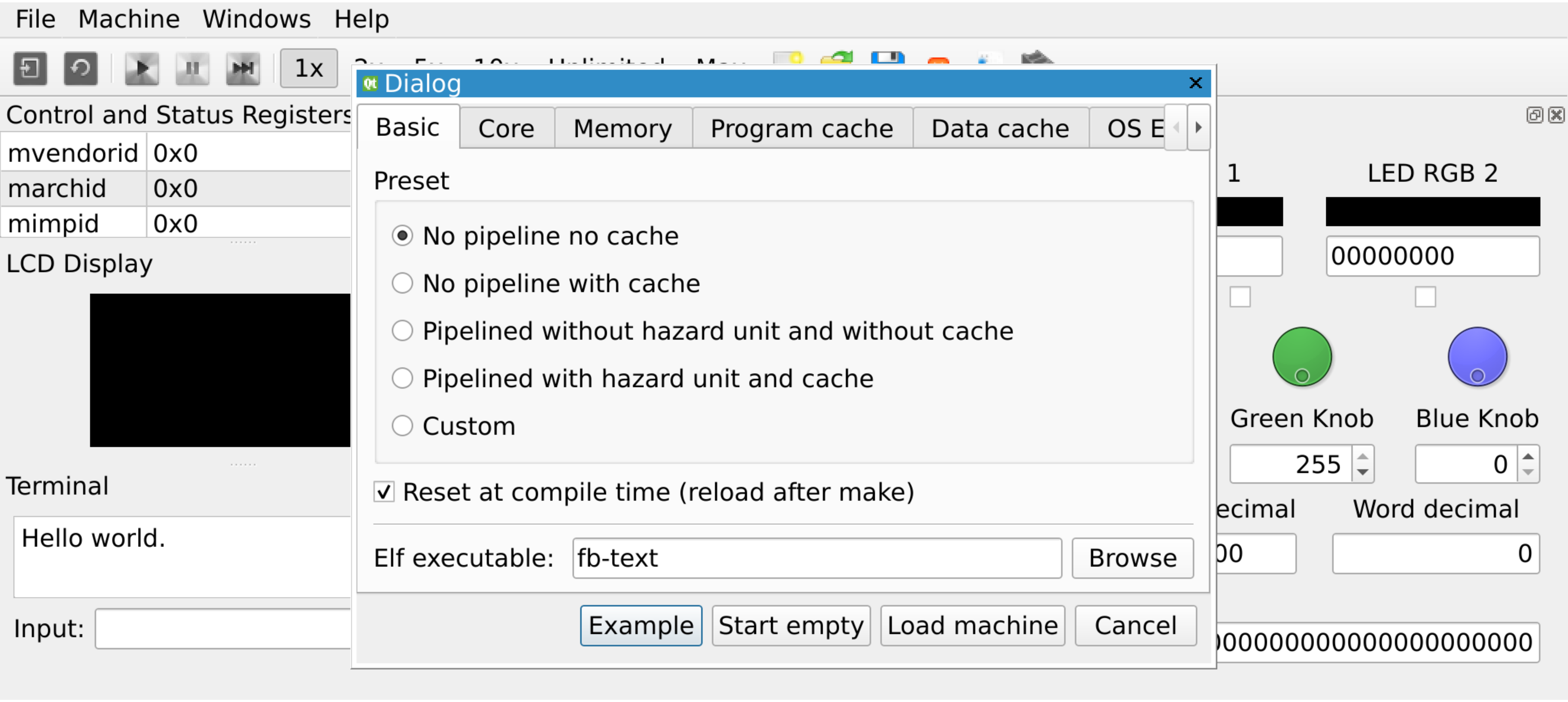


https://cw.fel.cvut.cz/b212/courses/b35apo/en/documentation/mz_apo/startv

# External Compilation

```
clang --target=riscv32
    -march=rv32g -nostdlib -static
    -fuse-ld=lld test.c -o test

riscv32-elf-gcc test.c -o test

riscv64-unknown-elf-gcc -ggdb -mabi=ilp32
    -march=rv32im test.c -o test
```

# Example: Frame Buffer 1



File Machine Windows Help
1x
Control and Status Registers
LCD Display
Terminal
Input:
LED RGB 2
Green Knob
Blue Knob
Word decimal
Dialog
Basic Core Memory Program cache Data cache OS E
Example Start empty Load machine Cancel

| | |
|---|---|
| mvendorid | 0x0 |
| marchid | 0x0 |
| mimpid | 0x0 |

```
Hello world.
```

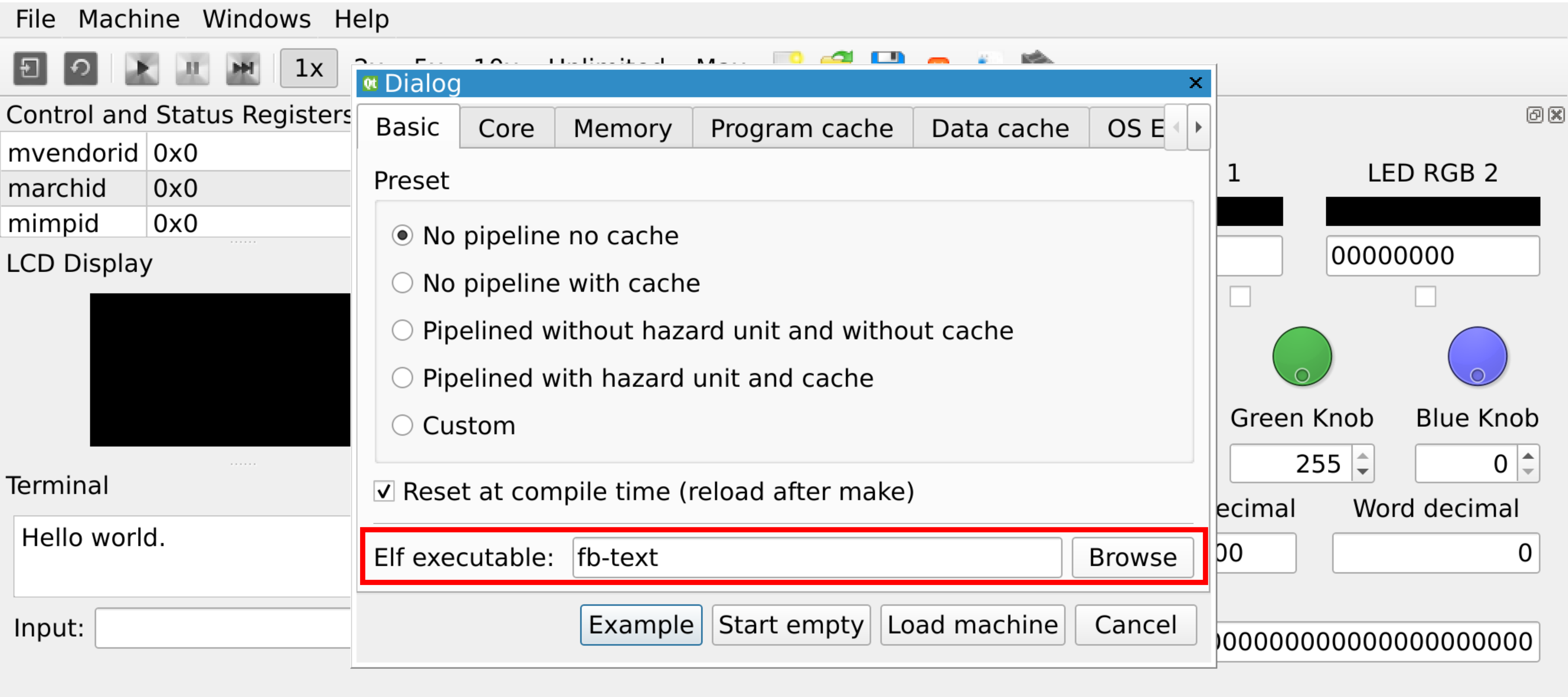
Preset

[● No pipeline no cache]
○ No pipeline with cache
○ Pipelined without hazard unit and without cache
○ Pipelined with hazard unit and cache
○ Custom

[✓] Reset at compile time (reload after make)

Elf executable: fb-text [Browse]

# Example: Frame Buffer 1



| | |
|---|---|
| mvendorid | 0x0 |
| marchid | 0x0 |
| mimpid | 0x0 |

Hello world.

Input:

Preset

- [✓ No pipeline no cache]
- No pipeline with cache
- Pipelined without hazard unit and without cache
- Pipelined with hazard unit and cache
- Custom

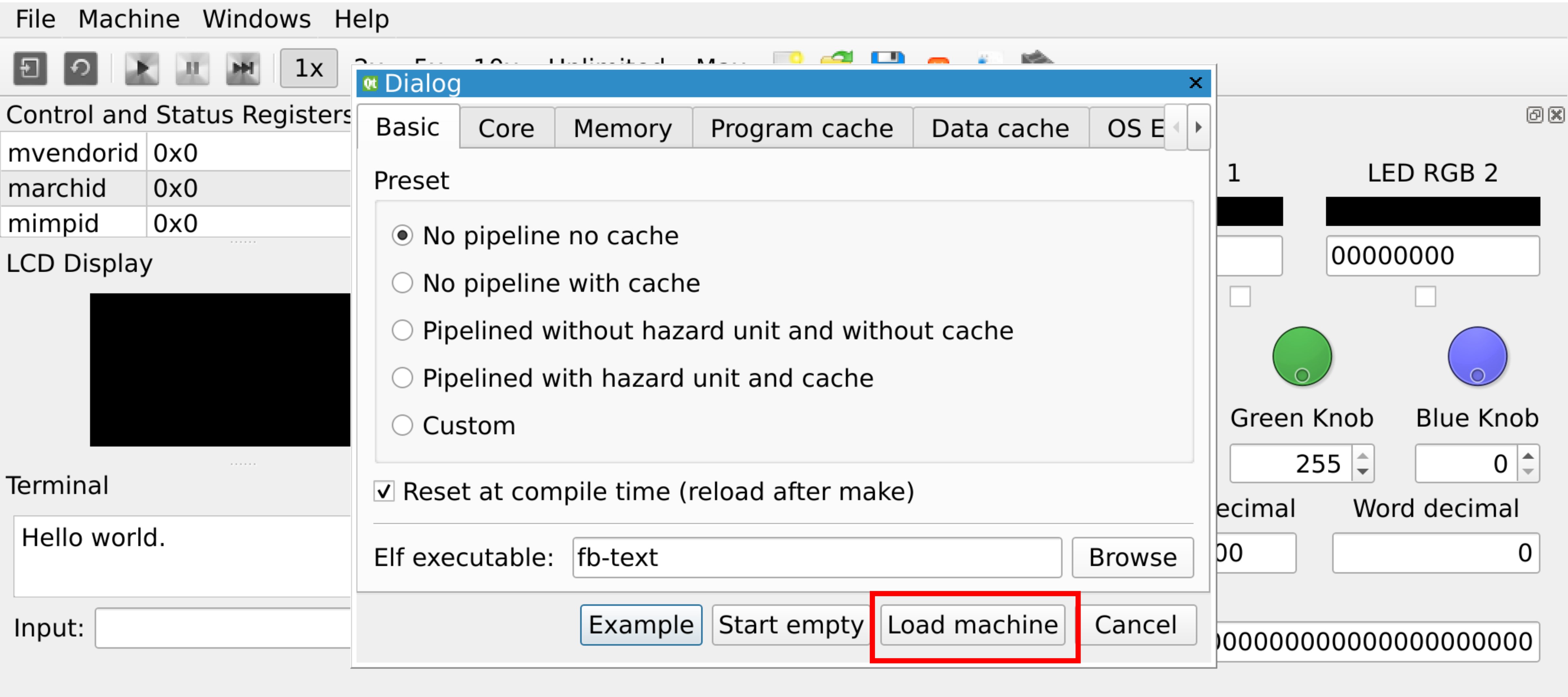
☑ Reset at compile time (reload after make)

Elf executable: fb-text

00000000

255 0

# Example: Frame Buffer 1



File Machine Windows Help

Control and Status Registers

| | |
|---|---|
| mvendorid | 0x0 |
| marchid | 0x0 |
| mimpid | 0x0 |

LCD Display

Terminal

```
Hello world.
```

Input:

Dialog

Basic | Core | Memory | Program cache | Data cache | OS E

Preset

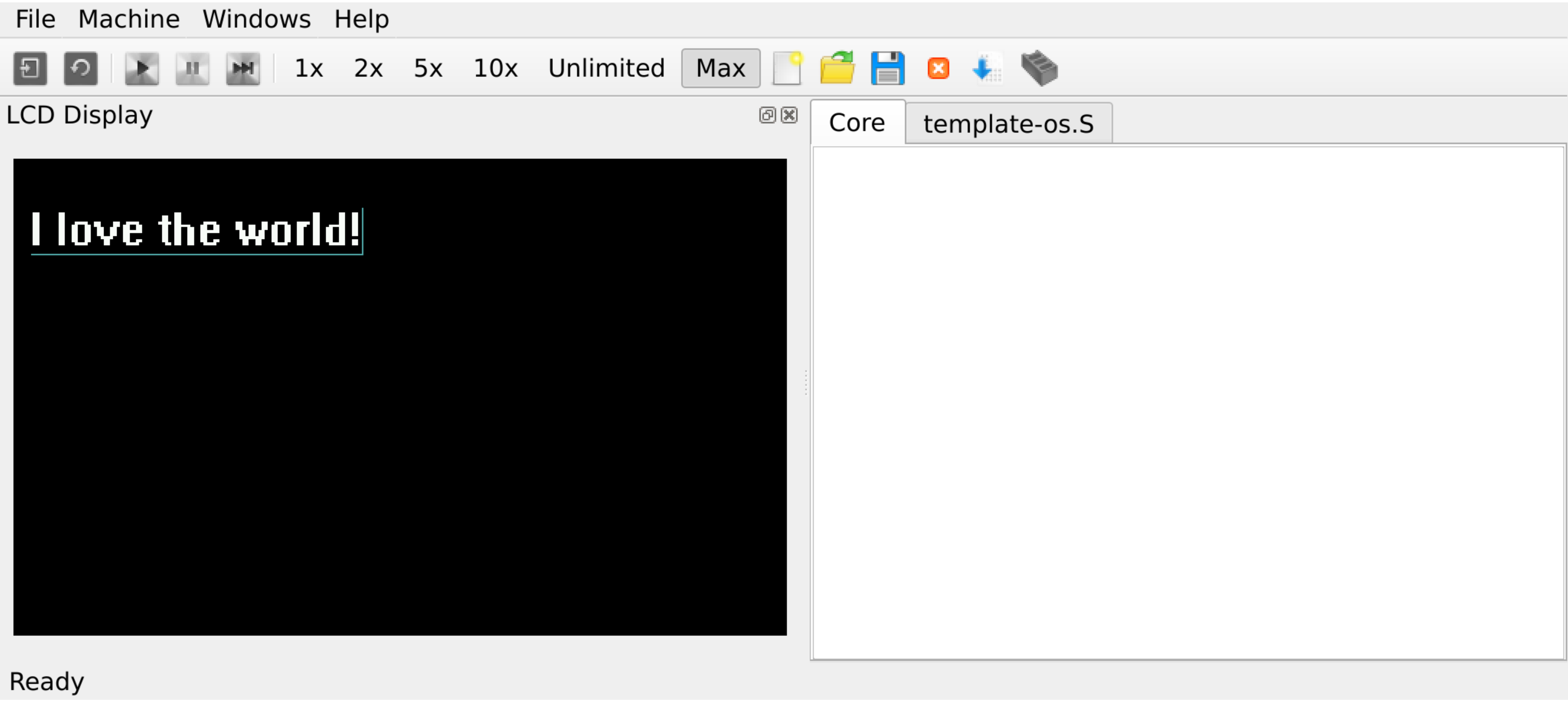
- [✓ No pipeline no cache]
- No pipeline with cache
- Pipelined without hazard unit and without cache
- Pipelined with hazard unit and cache
- Custom

☑ Reset at compile time (reload after make)

Elf executable: fb-text

Example | Start empty | Load machine | Cancel

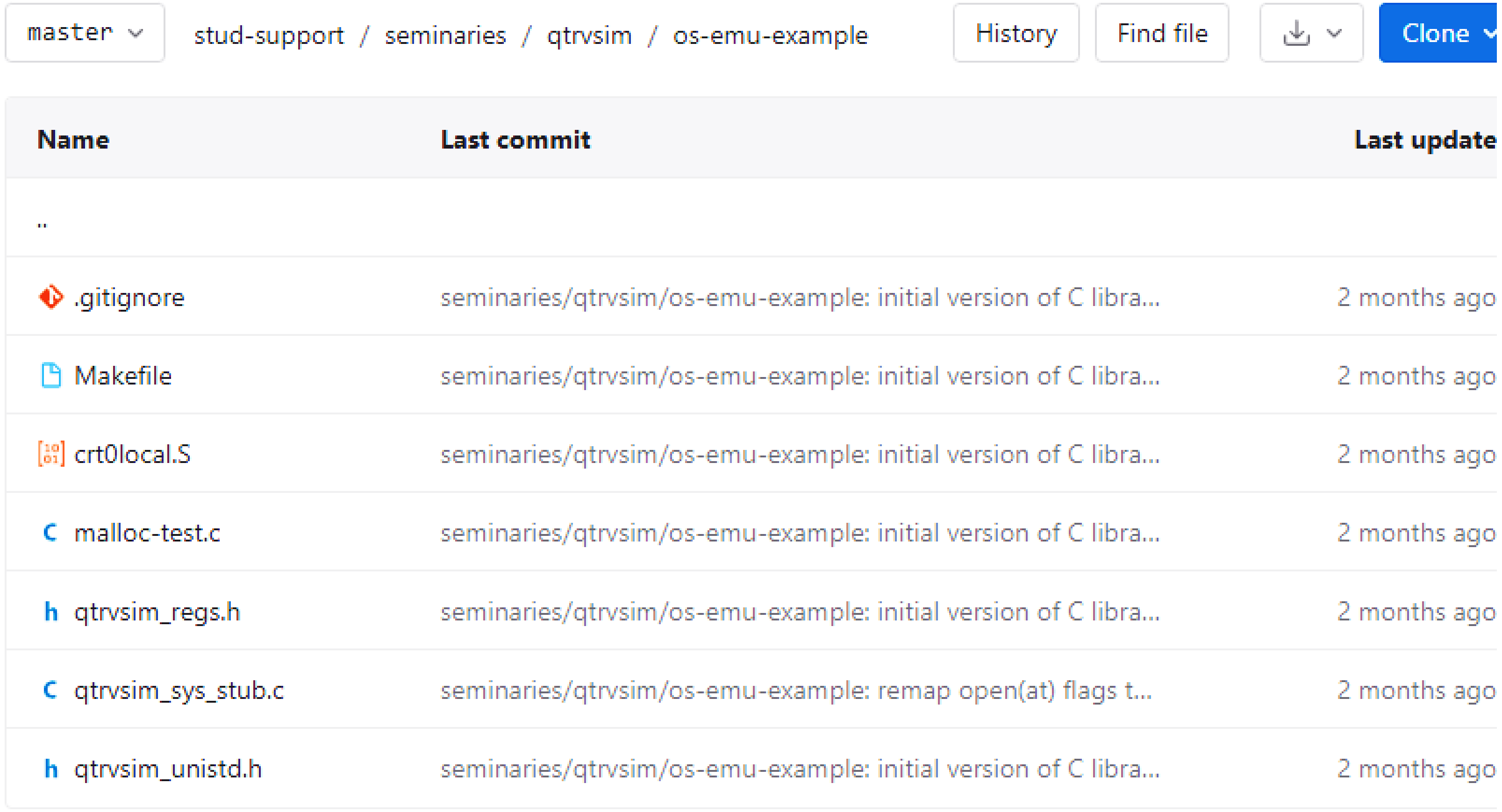
# Example: Frame Buffer 2



I love the world!

Ready

# NewLib Example Project

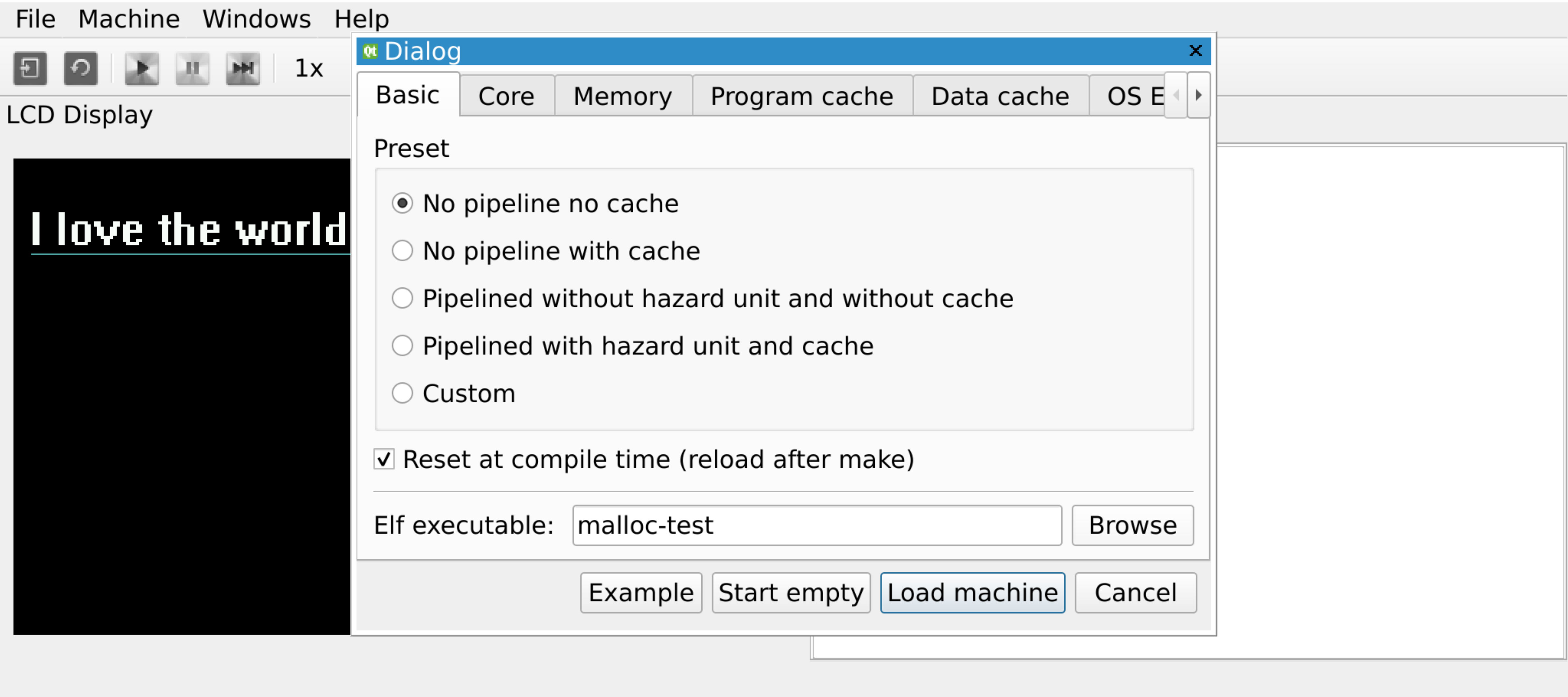


master | stud-support / seminaries / qtrvsim / os-emu-example | History | Find file | Clone

| Name | Last commit | Last update |
|---|---|---|
| .. | | |
| .gitignore | seminaries/qtrvsim/os-emu-example: initial version of C libra... | 2 months ago |
| Makefile | seminaries/qtrvsim/os-emu-example: initial version of C libra... | 2 months ago |
| crt0local.S | seminaries/qtrvsim/os-emu-example: initial version of C libra... | 2 months ago |
| malloc-test.c | seminaries/qtrvsim/os-emu-example: initial version of C libra... | 2 months ago |
| qtrvsim_regs.h | seminaries/qtrvsim/os-emu-example: initial version of C libra... | 2 months ago |
| qtrvsim_sys_stub.c | seminaries/qtrvsim/os-emu-example: remap open(at) flags t... | 2 months ago |
| qtrvsim_unistd.h | seminaries/qtrvsim/os-emu-example: initial version of C libra... | 2 months ago |

**https://gitlab.fel.cvut.cz/b35apo/stud-support/-/tree/master/seminaries/qtrvsim/os-emu-example**

# Example: NewLib Malloc 1



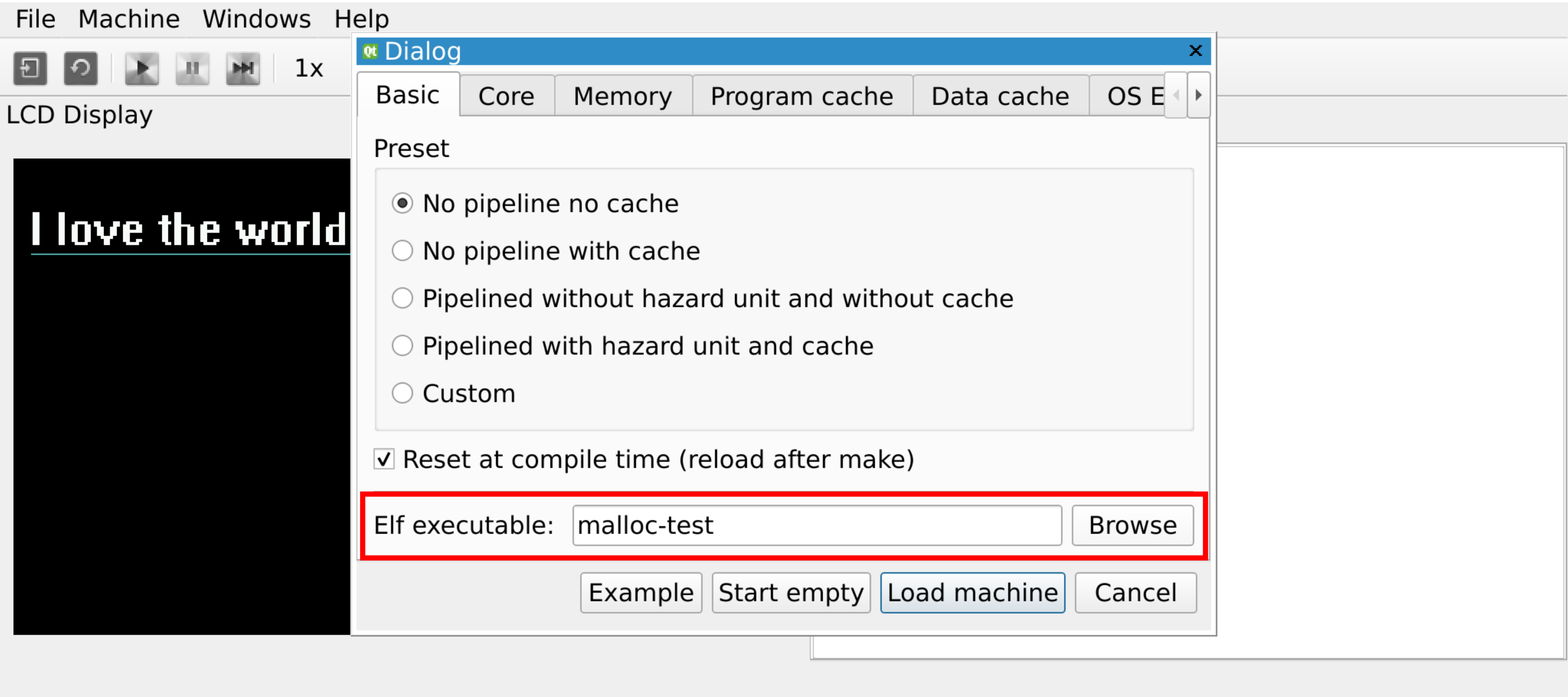
Preset

- [✓ No pipeline no cache]
- No pipeline with cache
- Pipelined without hazard unit and without cache
- Pipelined with hazard unit and cache
- Custom

✓ Reset at compile time (reload after make)

Elf executable: malloc-test

# Example: NewLib Malloc 1



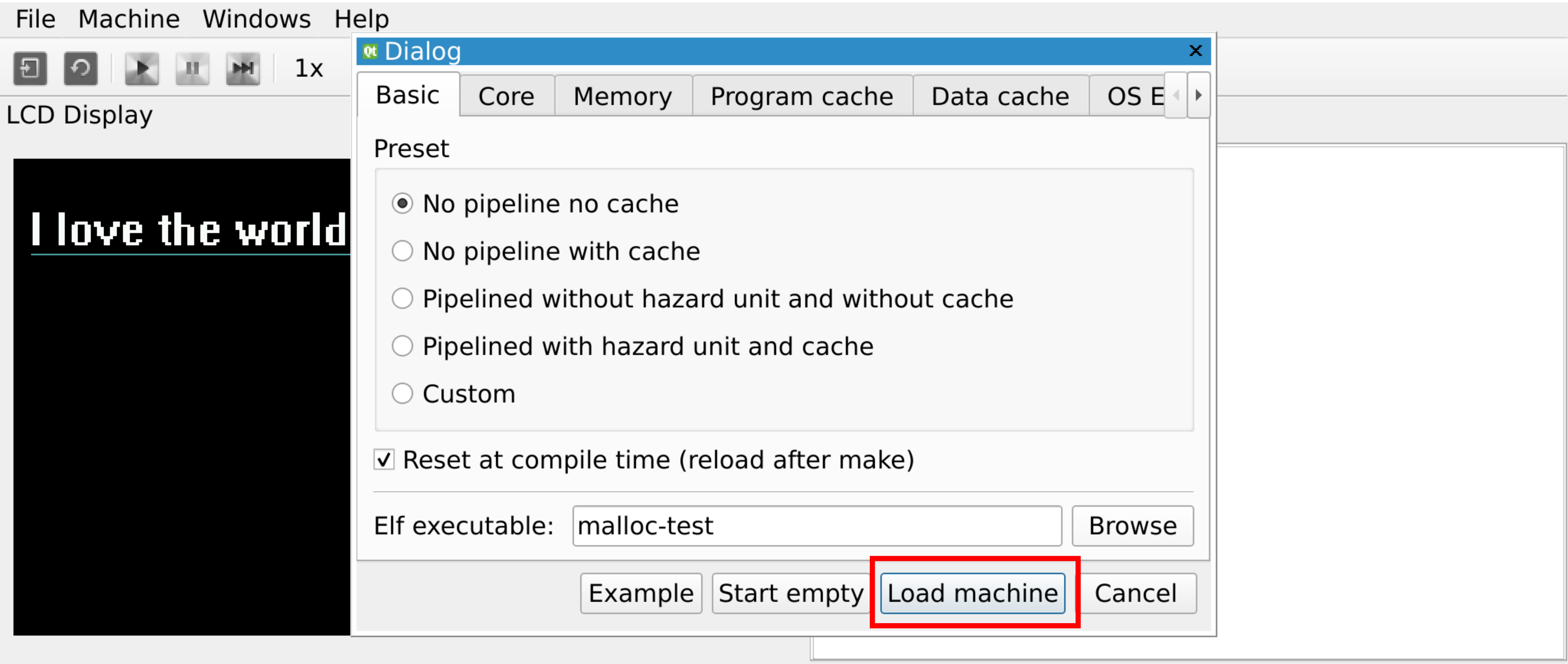
Preset

- [✓ No pipeline no cache]
- No pipeline with cache
- Pipelined without hazard unit and without cache
- Pipelined with hazard unit and cache
- Custom

✓ Reset at compile time (reload after make)

Elf executable: malloc-test

# Example: NewLib Malloc 1



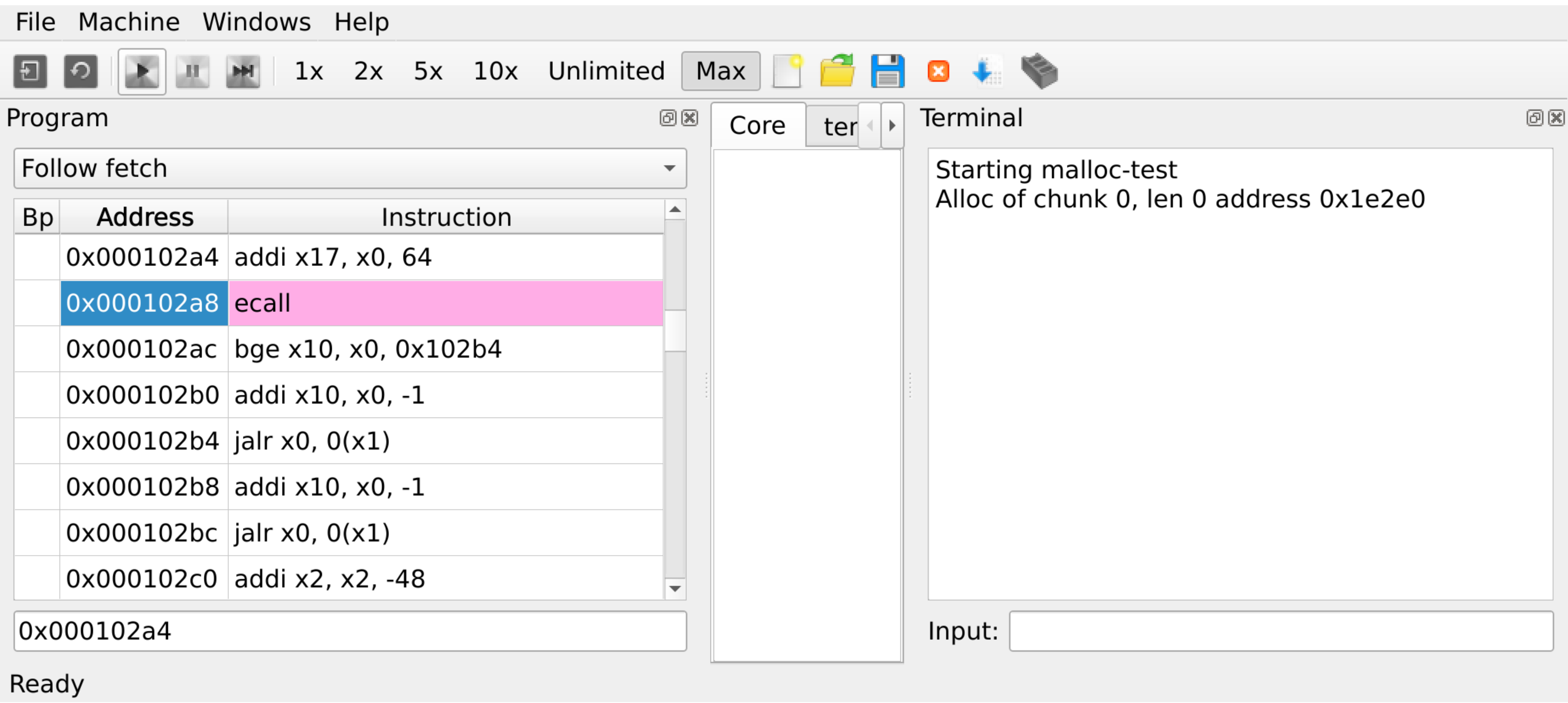
Preset

- [✓ No pipeline no cache]
- No pipeline with cache
- Pipelined without hazard unit and without cache
- Pipelined with hazard unit and cache
- Custom

✓ Reset at compile time (reload after make)

Elf executable: malloc-test

# Example: NewLib Malloc 2

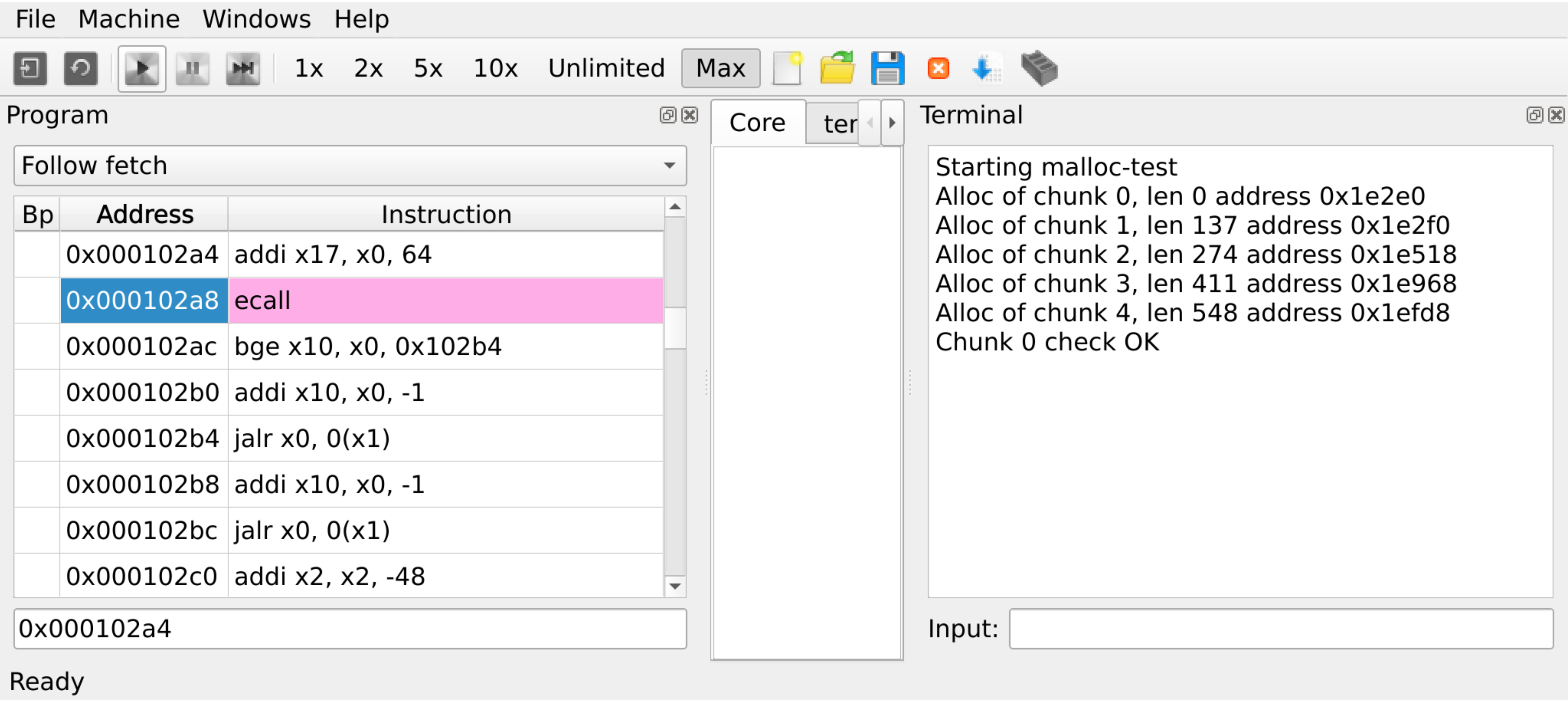


| Bp | Address | Instruction |
|---|---|---|
| | 0x000102a4 | addi x17, x0, 64 |
| | 0x000102a8 | ecall |
| | 0x000102ac | bge x10, x0, 0x102b4 |
| | 0x000102b0 | addi x10, x0, -1 |
| | 0x000102b4 | jalr x0, 0(x1) |
| | 0x000102b8 | addi x10, x0, -1 |
| | 0x000102bc | jalr x0, 0(x1) |
| | 0x000102c0 | addi x2, x2, -48 |

0x000102a4

```
Starting malloc-test
Alloc of chunk 0, len 0 address 0x1e2e0
```

## Example: NewLib Malloc 3

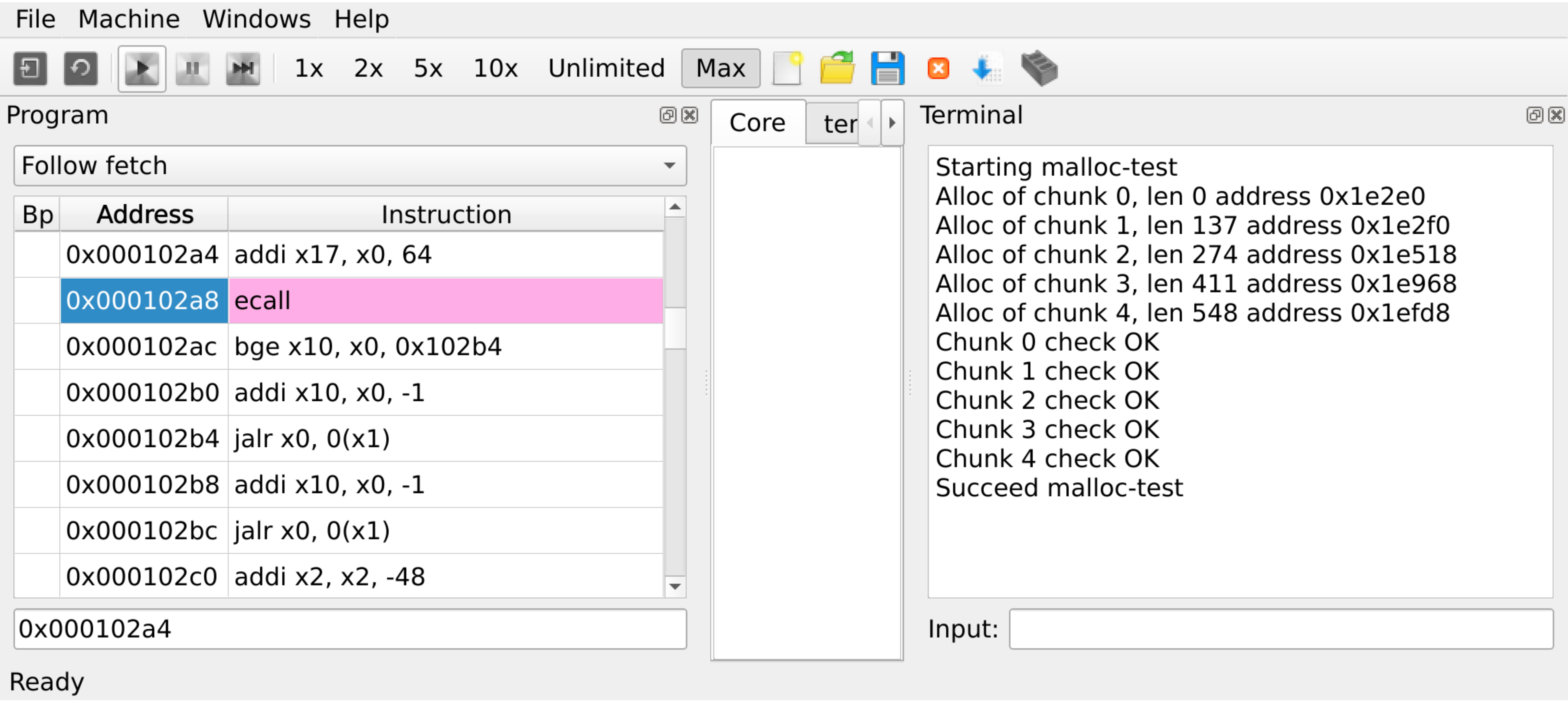


| Bp | Address | Instruction |
|---|---|---|
| | 0x000102a4 | addi x17, x0, 64 |
| | 0x000102a8 | ecall |
| | 0x000102ac | bge x10, x0, 0x102b4 |
| | 0x000102b0 | addi x10, x0, -1 |
| | 0x000102b4 | jalr x0, 0(x1) |
| | 0x000102b8 | addi x10, x0, -1 |
| | 0x000102bc | jalr x0, 0(x1) |
| | 0x000102c0 | addi x2, x2, -48 |

0x000102a4

```
Starting malloc-test
Alloc of chunk 0, len 0 address 0x1e2e0
Alloc of chunk 1, len 137 address 0x1e2f0
Alloc of chunk 2, len 274 address 0x1e518
Alloc of chunk 3, len 411 address 0x1e968
Alloc of chunk 4, len 548 address 0x1efd8
Chunk 0 check OK
```

# Example: NewLib Malloc 4



| Bp | Address | Instruction |
|---|---|---|
| | 0x000102a4 | addi x17, x0, 64 |
| | 0x000102a8 | ecall |
| | 0x000102ac | bge x10, x0, 0x102b4 |
| | 0x000102b0 | addi x10, x0, -1 |
| | 0x000102b4 | jalr x0, 0(x1) |
| | 0x000102b8 | addi x10, x0, -1 |
| | 0x000102bc | jalr x0, 0(x1) |
| | 0x000102c0 | addi x2, x2, -48 |

0x000102a4

```
Starting malloc-test
Alloc of chunk 0, len 0 address 0x1e2e0
Alloc of chunk 1, len 137 address 0x1e2f0
Alloc of chunk 2, len 274 address 0x1e518
Alloc of chunk 3, len 411 address 0x1e968
Alloc of chunk 4, len 548 address 0x1efd8
Chunk 0 check OK
Chunk 1 check OK
Chunk 2 check OK
Chunk 3 check OK
Chunk 4 check OK
Succeed malloc-test
```

## Conclusion

FAQ
Future Plans & Wishes
Calls for Cooperation
Materials

# FAQ

- **Cycle accuracy**

## FAQ

- Cycle accuracy
  - YES*

## FAQ

- **Cycle accuracy**
  - **YES***
- **RISC-V official tests compliant**
  - **YES** (NEW in 0.9.4), part of the CI

## FAQ

- **Cycle accuracy**
  - **YES***
- **RISC-V official tests compliant**
  - **YES** (NEW in 0.9.4), part of the CI
- **ISA extensions support**
  - **RV32IM, RV64IM** (CLI only), **Zicsr** (NEW in 0.9.4!)

## FAQ

- **Cycle accuracy**
    - **YES***
- **RISC-V official tests compliant**
    - **YES** (NEW in 0.9.4), part of the CI
- **ISA extensions support**
    - **RV32IM, RV64IM** (CLI only), **Zicsr** (NEW in 0.9.4!)
- Virtual memory / MMU
    - Not yet (**student work planned**)

## Future Plans & Wishes

- **Interupts**
- **Compressed ISA** support
  - Visualization
- **Instruction encoding detailed view**
- Run minimal **riscv64-linux-elf**
- **64bit visualization**
- **MMU – virtual memory**
- **Pipeline utilization graph**
- (Limited) **time-travel debugging** (step back)

## Calls for Cooperation

- **Teachers, Educational Institutions**
  - Use the simulator
  - Cooperate on open materials

## Calls for Cooperation

- **Teachers, Educational Institutions**
  - Use the simulator
  - Cooperate on open materials
- **Students, Developers**
  - Help develop the simulator
  - Great for final thesis

# Calls for Cooperation

- **Teachers, Educational Institutions**
  - Use the simulator
  - Cooperate on open materials
- **Students, Developers**
  - Help develop the simulator
  - Great for final thesis
- **Distribution maintainers**
  - Help us with packaging to official repositories

## How to Get It?

**Project Sources**

https://github.com/cvut/qtrvsim

**Windows, Linux, Mac**

https://github.com/cvut/qtrvsim/releases

**Ubuntu**

https://launchpad.net/~qtrvsimteam/+archive/ubuntu

**Suse, Fedora and Debian**

https://software.opensuse.org/download.html?project=home%3Ajdupak&package=qtrvsim

**AUR, Nixpkgs**

**Online version**

https://comparch.edu.cvut.cz/qtrvsim/app

GitHub

## Publications

**Graphical CPU Simulator with Cache Visualization**
Karel Kočí; Diploma Thesis
https://dspace.cvut.cz/bitstream/handle/10467/76764/F3-DP-2018-Koci-Karel-diploma.pdf

**Graphical RISC-V Architecture Simulator - Memory Model and Project Management**
Jakub Dupák; 2021; Bachelor Thesis
https://dspace.cvut.cz/bitstream/handle/10467/94446/F3-BP-2021-Dupak-Jakub-thesis.pdf

**Graphical RISC-V Architecture Simulator - Instructions Decode and Execution and OS Emulation**
Max Hollmann; 2021; Bachelor Thesis
https://dspace.cvut.cz/bitstream/handle/10467/96707/F3-BP-2021-Hollmann-Max-thesis.pdf

Dupák, J.; Píša, P.; Štepanovský, M.; Kočí, K.
**QtRVSim – RISC-V Simulator for Computer Architectures Classes**
In: embedded world Conference 2022. Haar: WEKA FACHMEDIEN GmbH, 2022. p. 775-778.
ISBN 978-3-645-50194-1.
https://comparch.edu.cvut.cz/publications/ewC2022-Dupak-Pisa-Stepanovsky-QtRvSim.pdf

## Teaching Materials

Faculty of Electrical Engineering

- B35APO - **Computer Architectures**
  - CZ + EN materials and videos
- BE4M35PAP - **Advanced Computer Architectures**
  - CZ materials and videos

Faculty of Information Technology

- BI-APS - **Architectures of Computer Systems**
  - EN materials

***https://comparch.edu.cvut.cz/***

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

# Thank you

***comparch.edu.cvut.cz***

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE