A4. Pipelined Processor

CSE261: Computer Architecture, Fall 2021

Hyungon Moon

Out: Nov 18, 2021

Due: Nov 30 2021 11:59pm

Goal

- Implement the pipelined mini-RISC-V processor that runs 5 instructions.
- Not supporting forwarding, for simplicity.
  - Still need to handle control hazard from branches.
- The five instructions
  - `addi, add, ld, sd, beq`

Vagrant and VirtualBox

- Vagrant allows you to easily create and access a virtual machine.
- Install Vagrant from

```
https://www.vagrantup.com/
```

- Vagrant (in this class) need VirtualBox. Install from

```
https://www.virtualbox.org/
```

- Should work on Ubuntu, Mac or Windows.
- May not work inside another virtual machine.

Initializing VM (Mac, Ubuntu (Linux))

- Use terminal to enter the `hw1` directory
- Make sure that you are seeing `Vagrantfile` in the directory
- Create and boot VM

```
vagrant up
```

- Connect to the VM

```
vagrant ssh
```

- From V2 of the assignment, the `hw1` directory is automatically synced with the `hw` directory

Initializing VM (Windows)

- Use terminal to enter the `hw1` directory
- Make sure that you are seeing `Vagrantfile` in the directory
- Create and boot VM

```
vagrant.exe up
```

- Connect to the VM

```
vagrant.exe ssh
```

- From V2 of the assignment, the `hw1` directory is automatically synced with the `hw` directory

SBT

- We will use a complex set of tools enabling us to write in Chisel.
- We don't need to care about them: SBT does the dirty job.
  - Located in `hw/sbt` or `sbt`.
- You will use three commands, referring to the lecture slides.
  - To run the Main object (and test the core in this assignment)

```
sbt/bin/sbt run
```

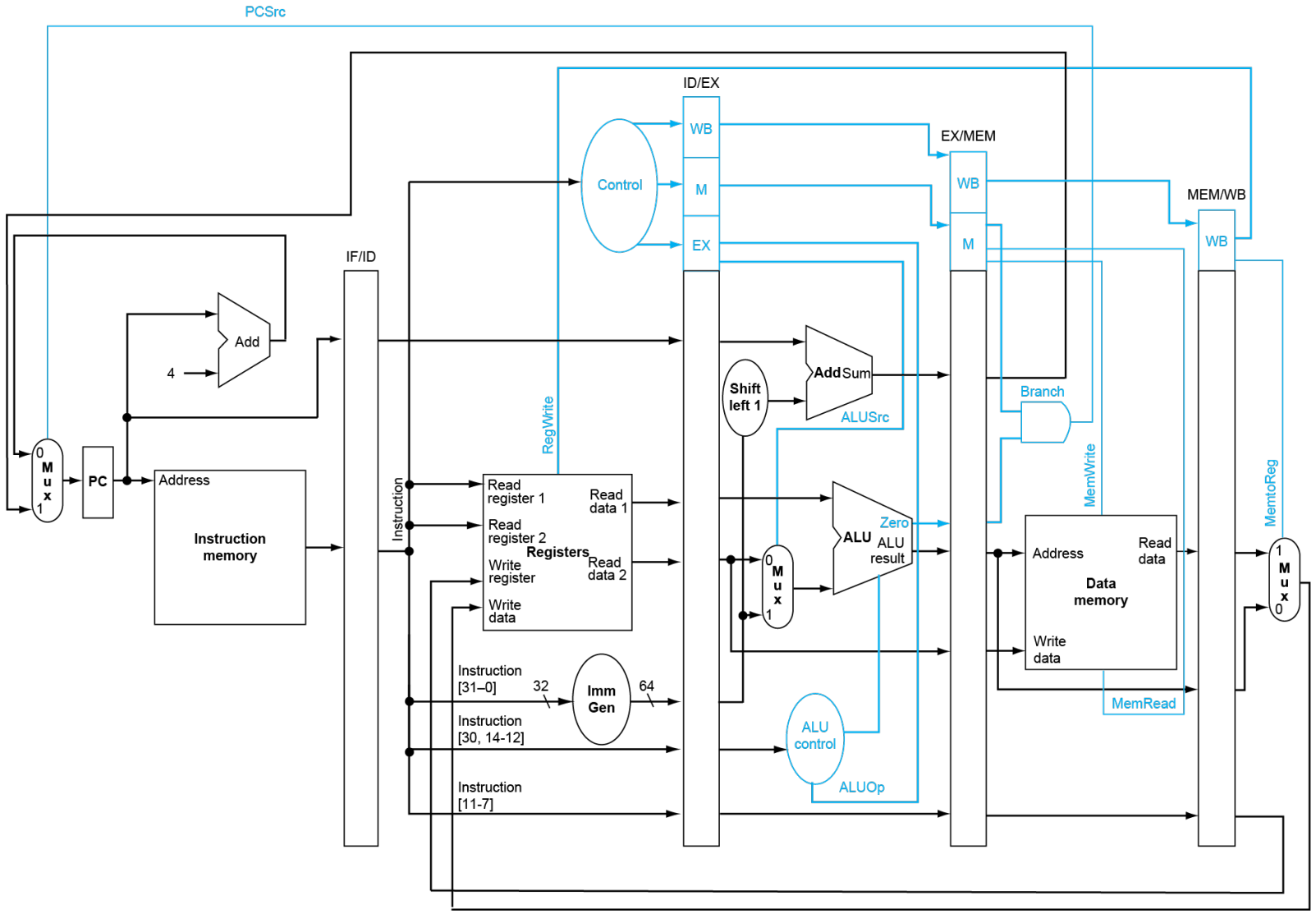
SBT trouble shooting

- When running sbt for the first time, you may see this message

```
java.io.IOException: org.scalasbt.ipcsocket.NativeErrorException:
[1] Operation not permitted
Create a new server? y/n (default y)
```

- Just press enter to go for the default value.

Goal: implementing a pipelined processor



Skeleton: Hw4Base (1/2)

- BaseSystem is the skeleton containing core and two memories.

```
class BaseSystem(prog: Array[String] = Array("x13", "x80", "x30", "x00")) extends
Module {
  val io = IO(new Bundle{
    val trace = Output(new Trace())
  })

  // Submodules
  val core = Module(new Core())
  val imem = Module(new InstructionROM(prog))
  val dmem = Module(new ReadWriteMem())

  // Connect instruction memory
  imem.io.addr := core.io.imem_addr
  core.io.imem_insn := imem.io.dataOut
```

Skeleton: Hw4Base (2/2)

- BaseSystem is the skeleton containing core and two memories.

```
  // Connect data memory
  dmem.io.addr := core.io.dmem_addr
  dmem.io.dataIn := core.io.dmem_wdata
  dmem.io.write := core.io.dmem_write
  dmem.io.read := core.io.dmem_read
  core.io.dmem_rdata := dmem.io.dataOut

  // Trace interface
  val reg_dmem_read = RegNext(core.io.dmem_read)
  io.trace.pc := core.io.imem_addr
  io.trace.fetched_insn := imem.io.dataOut
  io.trace.wb_insn := core.io.wb_insn
…

}
```

Skeleton: Core (2/1)

- The interface is similar to the Hw3

```
class Core extends Module {
  val io = IO(new Bundle {
    //val reset = Input(Bool())
    val imem_addr = Output(UInt(64.W))
    val imem_insn = Input(UInt(32.W))
    val dmem_addr = Output(UInt(64.W))
    val dmem_write = Output(Bool())
    val dmem_read = Output(Bool())
    val dmem_wdata = Output(UInt(64.W))
    val dmem_rdata = Input(UInt(64.W))

    val wb_insn = Output(UInt(32.W))     // for log generation.
    val halted = Output(Bool())

  })
…
```

Skeleton: Core (2/2)

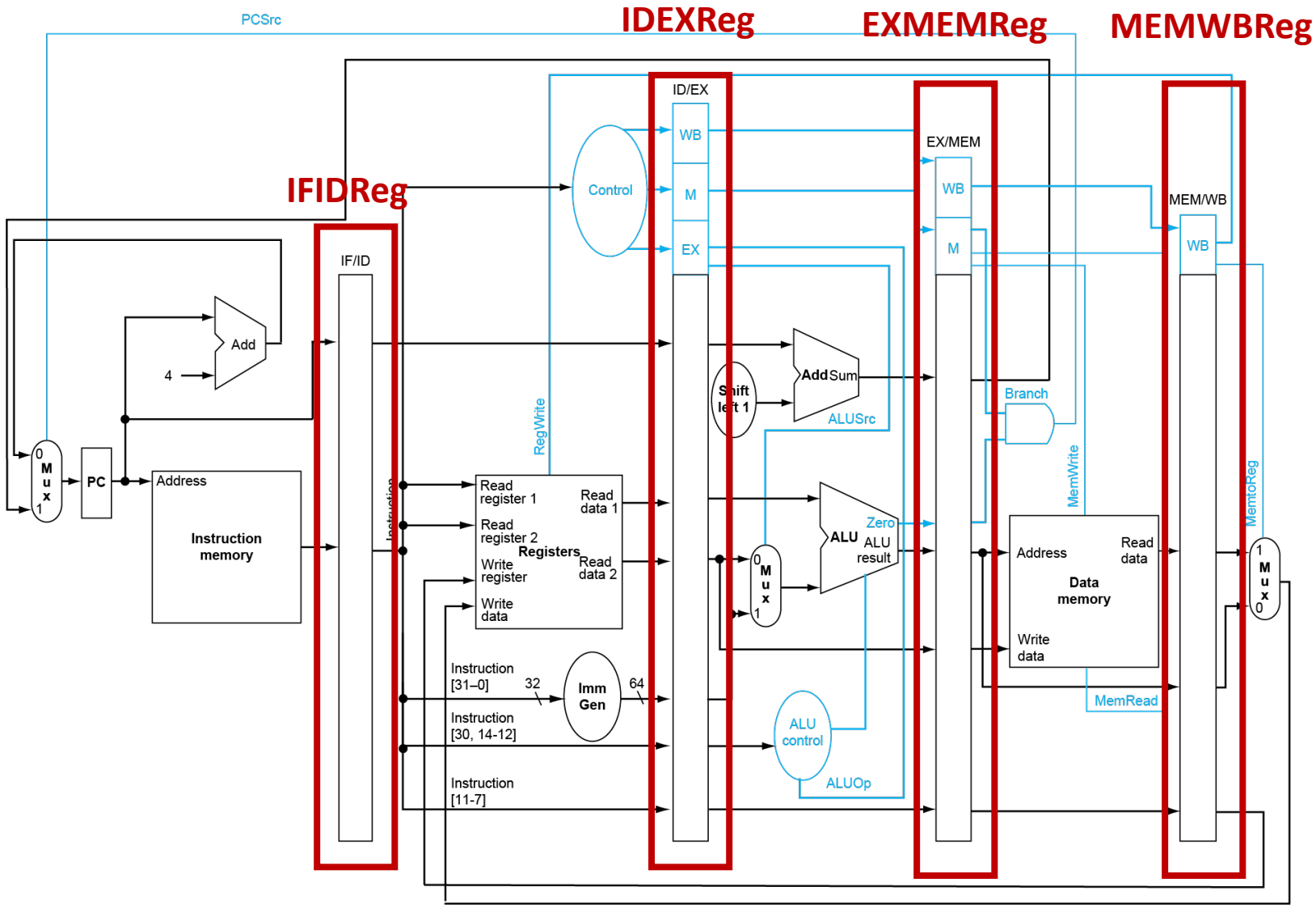
- Pipeline registers are predefined.

```
// pipeline registers

 val ifid_reg = Module(new IFIDReg())
 val idex_reg = Module(new IDEXReg())
 val exmem_reg = Module(new EXMEMReg())
 val memwb_reg = Module(new MEMWBReg())

 // combinational modules
 val control = Module(new ControlModule())
 val decoder = Module(new Decoder())
 val immGen = Module(new ImmGen())
 val regfile = Module(new RegFile())
 val aluControl = Module(new ALUControl())
 val alu = Module(new ALU())
```

Skeleton: pipeline registers (1/2)



Skeleton: pipeline registers (2/2)

- Defined in `src/main/scala/Hw4Pipeline.scala`
- Each register sets is defined with a Bundle and Module

```
class IFID extends Bundle {
  val pc = UInt(64.W)
  val insn = UInt(32.W)
  val valid = Bool()
…
}
```

```
class IFIDReg extends Module {
  val io = IO(new Bundle{
    val in = Input(new IFID())
    val stall = Input(Bool())
    val out = Output(new IFID())
  })

  val reg = Reg(new IFID())
  reg := io.in
  io.out := reg
  val valid_reg = RegInit(false.B)
  valid_reg := io.in.valid
  io.out.valid := Mux(io.stall, false.B, valid_reg)
}
```

Task Overview

- The test driver compiles test programs and runs on the processor.
- Tests in `src/main/scala/Tests.scala` defines four tests.
- Goal is to implement the register file and a processor running the four tests correctly.
  - Task 2: addinh
  - Task 3: addnh
  - Task 4: load and store (ldsdnh)
  - Task 5: branch (beqnh)
- Note: test cases do not cause data hazard.

Test Infra (1/3)

- Running programs (use `sbt run`)
  - `src/main/scala/UniRV64IDriver.scala`: compiles and runs program
    - No need to understand.
  - `sw/test` contains the four tests.
    - `addinh.s, addnh.s, ldsdnh.s, beqnh.s`
  - `src/main/scala/Tests.scala` contains the reference outputs
- Correct output

```
test results:
tested addi (addi.s): passed
tested add (add.s): passed
tested ldsd (ldsd.s): passed
tested beq (beq.s): passed
```

Test Infra (2/3)

- Testing individual cases
  - ➢ You can test only one case by (e.g., `addi.s`)

```
sbt/bin/sbt "run addinh.s"
```

  - ➢ Use the corresponding test names.
- After running each test, two file could help you debugging.
  - ➢ `sw/test/addinh.d`: shows the code layout.
  - ➢ `addinh.log`: the log generated from the test.

Test Infra (3/3)

- Dump example: `sw/test/addinh.d`

```
sw/test/addinh.elf:     file format elf64-littleriscv


Disassembly of section .text:

0000000000000000 <_start>:
   0:   01008093            addi    x1,x1,16
   4:   00000013            nop
   8:   00000013            nop
   c:   00000013            nop
  10:   00103023            sd  x1,0(x0) # 0 <_start>
  14:   00005073            csrwi   ustatus,0
```

Test Infra (4/4)

- Log Example: `addinh.log`

cycle	pc	fetched_insn	wb_insn	write	wdata	addr	read	rdata	halted
0	0	1008093	0	0	0	0	0	0	0
1	4	13	0	0	0	0	0	0	0
2	8	13	0	0	0	0	0	0	0
3	c	13	0	0	0	0	0	0	0
4	10	103023	1008093	0	0	0	0	0	0
5	14	5073	13	0	0	0	0	0	0
6	18	53	13	0	0	0	0	0	0
7	1c	53	13	1	10	0	0	0	0
8	20	53	103023	0	0	0	0	0	0
9	24	53	5073	0	0	0	0	0	1

Debug messages

- The skeleton also prints the pipeline register contents for debugging.
- You can also add your own prints for debugging.

```
Log @ 28
IFID: (78, 13, 1)
IDEX: (74, 13, 1) ((1, 0, 0, 0; 0, 3, 0, 0; 0, 0, 0)) (0, 0, 0, 0, 0, 0)
EXMEM: (70, 5073, 1) ((0, 0, 0, 0; 0, 3, 0, 0; 0, 0, 0)) (0, 112, 0, 0, 0)
MEMWB: (6c, 13, 1) ((1, 0, 0, 0; 0, 3, 0, 0; 0, 0, 0)) (0, 0)
------------------------------------------------------------
Register content: Vec(0, 516, 516, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
```

- Please refer to `src/main/scala/Hw4Pipeline.scala`

```
override def toPrintable: Printable = {
    p"IFID: (${Hexadecimal(pc)}, ${Hexadecimal(insn)}, ${valid})"
  }
```

Task 1: addinh

- Test program

```
.global _start
_start:
  addi   x1, x1, 0x10
  nop
  nop
  nop
  sd     x1, 0(x0)
  csrrwi x0, 0, 0
```

Task 1: addinh

- Expected result

cycle	pc	fetched_insn	wb_insn	write	wdata	addr	read	rdata	halted
0	0	1008093	0	0	0	0	0	0	0
1	4	13	0	0	0	0	0	0	0
2	8	13	0	0	0	0	0	0	0
3	c	13	0	0	0	0	0	0	0
4	10	103023	1008093	0	0	0	0	0	0
5	14	5073	13	0	0	0	0	0	0
6	18	53	13	0	0	0	0	0	0
7	1c	53	13	1	10	0	0	0	0
8	20	53	103023	0	0	0	0	0	0
9	24	53	5073	0	0	0	0	0	1

Task 2: addnh

- Test program

```
.global _start
_start:
  addi   x1, x0, 0x10
  addi   x2, x0, 0x20
  nop
  nop
  nop
  add   x3, x1, x2
  nop
  nop
  nop
  ld     x1, 0(x3)
  csrrwi x0, 0, 0
```

Task 2: addinh

- Expected result

cycle	pc	fetched_insn	wb_insn	write	wdata	addr	read	rdata	halted
0	0	1000093	0	0	0	0	0	0	0
1	4	2000113	0	0	0	0	0	0	0
2	8	13	0	0	0	0	0	0	0
3	c	13	0	0	0	0	0	0	0
4	10	13	1000093	0	0	0	0	0	0
5	14	2081b3	2000113	0	0	0	0	0	0
6	18	13	13	0	0	0	0	0	0
7	1c	13	13	0	0	0	0	0	0
8	20	13	13	0	0	0	0	0	0
9	24	1b083	2081b3	0	0	0	0	0	0
10	28	5073	13	0	0	0	0	0	0
11	2c	53	13	0	0	0	0	0	0
12	30	53	13	0	0	30	1	0	0
13	34	53	1b083	0	0	0	0	0	0
14	38	53	5073	0	0	0	0	0	1

Task 3: ldsdnh

- Test program

```
.global _start
_start:
  addi   x1, x0, 0x204
  nop
  nop
  nop
  sd     x1, 8(x0)
  nop
  ld     x2, 8(x0)
  ld     x3, 8(x1)
  nop
  nop
  sd     x1, 0(x2)
  csrrwi x0, 0, 0
```

Task 3: ldsdnh

- Expected result

cycle	pc	fetched_insn	wb_insn	write	wdata	addr	read	rdata	halted
0	0	20400093	0	0	0	0	0	0	0
1	4	13	0	0	0	0	0	0	0
2	8	13	0	0	0	0	0	0	0
3	c	13	0	0	0	0	0	0	0
4	10	103423	20400093	0	0	0	0	0	0
5	14	13	13	0	0	0	0	0	0
6	18	803103	13	0	0	0	0	0	0
7	1c	80b183	13	1	204	8	0	0	0
8	20	13	103423	0	0	0	0	0	0
9	24	13	13	0	0	8	1	0	0
10	28	113023	803103	0	0	20c	1	204	0
11	2c	5073	80b183	0	0	0	0	20c	0
12	30	53	13	0	0	0	0	0	0
13	34	53	13	1	204	204	0	0	0
14	38	53	113023	0	0	0	0	0	0
15	3c	53	5073	0	0	0	0	0	1

Task 4: beqnh

- Test program

```
.global _start
_start:
  addi   x1, x0, 0x204
  addi   x2, x0, 0x204
  nop
  nop
  nop
  ld     x1, 0(x1)
  ld     x2, 0(x2)
  nop
  nop
  nop
  nop
  beq    x1, x2, target
  nop
  nop
```

```
nop
  nop
  csrrwi x0, 0, 0
  nop
  nop
  sd     x1, 0(x0)
target:
  nop
  nop
  sd     x1, 0(x0)
  beq    x1, x0, target2
  nop
  nop
  nop
  nop
  csrrwi x0, 0, 0
```

```
target2:
  nop
  nop
  nop
  nop
  sd     x0, 0(x0)
  nop
  nop
  csrrwi x0, 0, 0
```

Task 4: beqnh

- Expected result

cycle	pc	fetched_insn	wb_insn	write	wdata	addr	read	rdata	halted
0	0	20400093	0	0	0	0	0	0	0
1	4	20400113	0	0	0	0	0	0	0
2	8	13	0	0	0	0	0	0	0
3	c	13	0	0	0	0	0	0	0
4	10	13	20400093	0	0	0	0	0	0
5	14	b083	20400113	0	0	0	0	0	0
6	18	13103	13	0	0	0	0	0	0
7	1c	13	13	0	0	0	0	0	0
8	20	13	13	0	0	204	1	0	0
9	24	13	b083	0	0	204	1	204	0
10	28	13	13103	0	0	0	0	204	0
11	2c	2208263	13	0	0	0	0	0	0
12	30	13	13	0	0	0	0	0	0
13	34	13	13	0	0	0	0	0	0
14	34	13	13	0	0	0	0	0	0
15	50	13	2208263	0	0	0	0	0	0
16	54	13	0	0	0	0	0	0	0

Task 4: beqnh

- Expected result

17	58	103023	0	0	0	0	0	0	0
18	5c	8c63	13	0	0	0	0	0	0
19	60	13	13	0	0	0	0	0	0
20	64	13	13	1	204	0	0	0	0
21	64	13	103023	0	0	0	0	0	0
22	68	13	8c63	0	0	0	0	0	0
23	6c	13	0	0	0	0	0	0	0
24	70	5073	0	0	0	0	0	0	0
25	74	13	13	0	0	0	0	0	0
26	78	13	13	0	0	0	0	0	0
27	7c	13	13	0	0	0	0	0	0
28	80	13	5073	0	0	0	0	0	1