Intro to Processor Architecture

Mid Evaluation Project Report

Topic: Designing a processor using sequential and pipeline architectures.

Team Name: **Disconnected**

Team Members:

Pranav Manu (2020112019)

Rohan Reddy Bodgam (2020102039)

Progress made:

Sequential Hardware Implementation:

The sequential hardware implementation consists of 6 stages.

1) Fetch
2) Decode
3) Execute
4) Memory
5) Write-Back
6) PC update

Fetch:

- The fetch block is the first stage in sequential implementation which reads the given instruction and sends it to the next stage in the required input form.
- Each instruction given will be in the form of 10 bytes from the instruction memory according to the value of the PC in which the first byte is always read as he instruction code and instruction function each read from 4 MSB (bits at position 7 to 4) add 4 LSB (bits at position 3 to 0 where the byte is read from 7 to 0) of the first byte respectively and it is determined to be valid or not depending on the value of the instruction code given. Using the icode, the required length of the instruction is known.

- The second byte is valC or addresses rA and rB depending on icode, and the value of need_regids is set.
- There will also be a 64-bit value PC which gets incremented according to the value of need_regids (if rA and rB are required to carry out the instruction) and need_valC(if valC is required to carry out the instruction) represented in mathematical terms as valP = PC + need_regids + 8*need_valC.

Decode:

- The decode stage is done in the register file (RegFile module) which takes the inputs icode, rA and rB.
- The register file stores fifteen registers and the output of the decode stage will be valA and valB whose values will the values stored in register rA and register rB respectively in most of the instructions given but in some case it may be equal to the register with stack pointer which is 4. Combinational circuit decides the value of srcA, srcB, dstM and dstE.
- The valA and valB will be given as outputs according to the values of icode, rA and rB and the values stored in the 15 registers present in the register file.

Execute:

- The execute stage is the place where the ADD, SUB, AND, XOR operations take place in the ALU according to the values of icode, valA, valB and valC and for some values of icode the condition code is calculated which some of the next stages depend on.
- The output of the execute block is valE and condition code, which is set as a side effect of arithmetic or logical operation.
- ALU_fun wrapper module is used to initialize the cnd depending on the icode value. The condition code registers are updated only at positive edge of clock.

Memory:

- In memory block, the values of valA, valP, valE and valB are taken as inputs and they are written to the memory or read from the memory at certain indexes to get valM as the output.
- Each value in the memory is stored in terms of 8 bits which means the 64-bit values are stored in 8 successive cells of 8 bits each when being written

to the memory and likewise when the memory is being read to get valM, it reads 8 successive cells of 8 bits each and transforms it to a 64-bit value.

- Data is written onto the memory only at positive edge of the clock cycle.
- A data memory wrapper module is used to decide whether reading is required or writing, and from which address. The data in memory is stored in little endian format

Write-back:

In write-back stage, the values of valM or valE will be written/updated in registers rA or rB or 4(stack-pointer) at positive edge of the clock, depending on the instruction code given to execute. The writeback happens in RegFile module

PC update:

After every instruction is completed, the PC gets updated in the PC update stage which will be valP, valC or valM depending on the instruction code given and the condition code acquired in the execute stage for some instructions.

A wrap module is used to initialize all the other modules and provide each module with the correct input signals

Instruction that are given to the processor

1. halt
2. nop
3. cmovXX (XX depends on the function code we give)
4. irmovq
5. rmmovq
6. mrmovq
7. opq
8. jXX
9. call
10. ret
11. pushq
12. popq

Instruction given and output of the instructions:

Set 1:

irmovq 5, 11

irmovq 10, 10

opq 11, 10

irmovq 4294440496 6

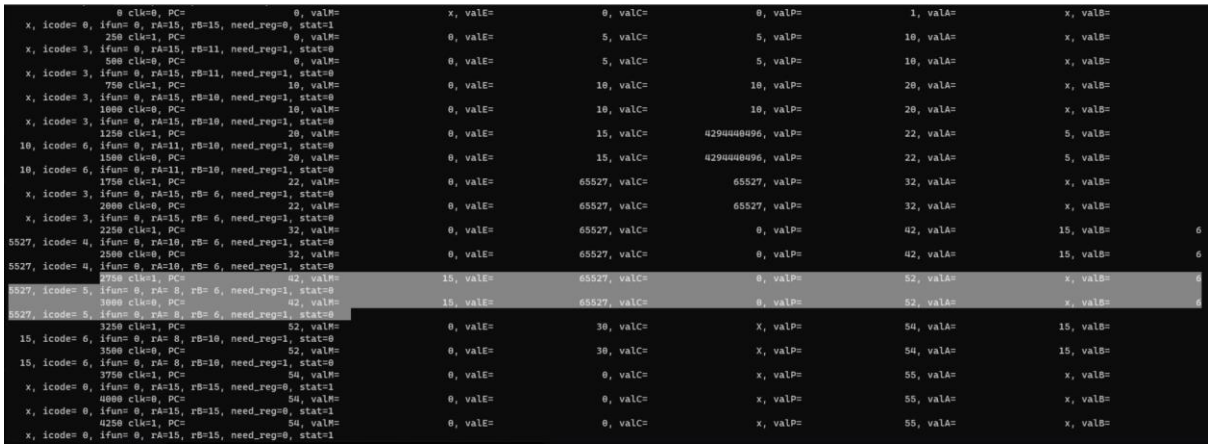
rmmovq 10 0(6)

mrmovq 0(6) 8

opq 8 10

halt

Output:



Set 2:

irmovq 5, 11

irmovq 10, 10

opq 11, 10

irmovq 0, 6

rmmovq 10, 0(6)

irmovq 10, 4

pushq 10

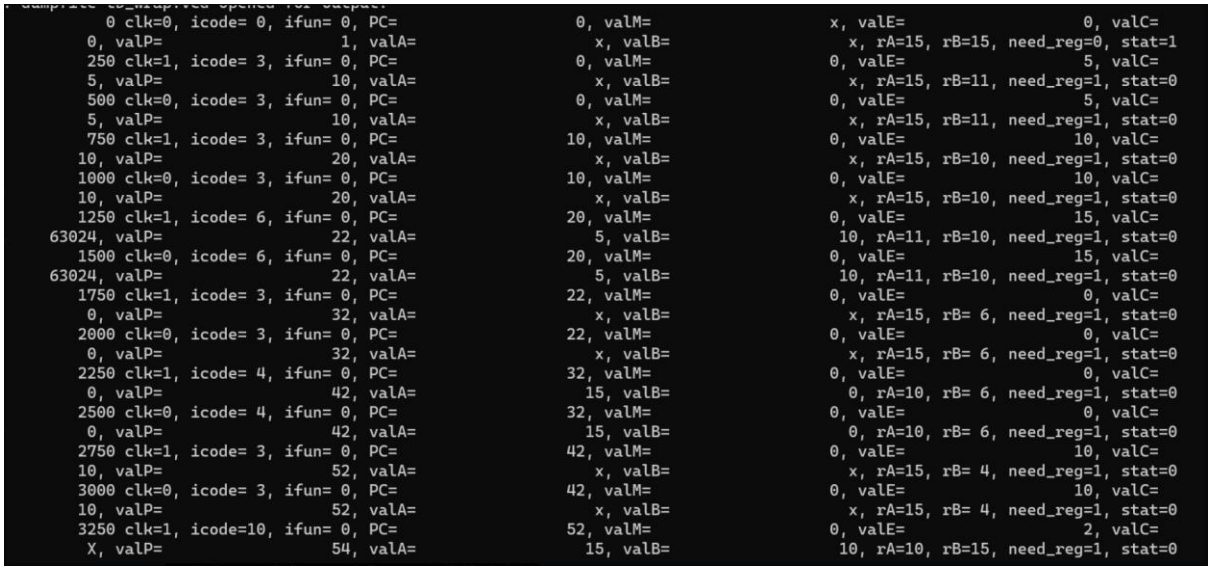
popq 10

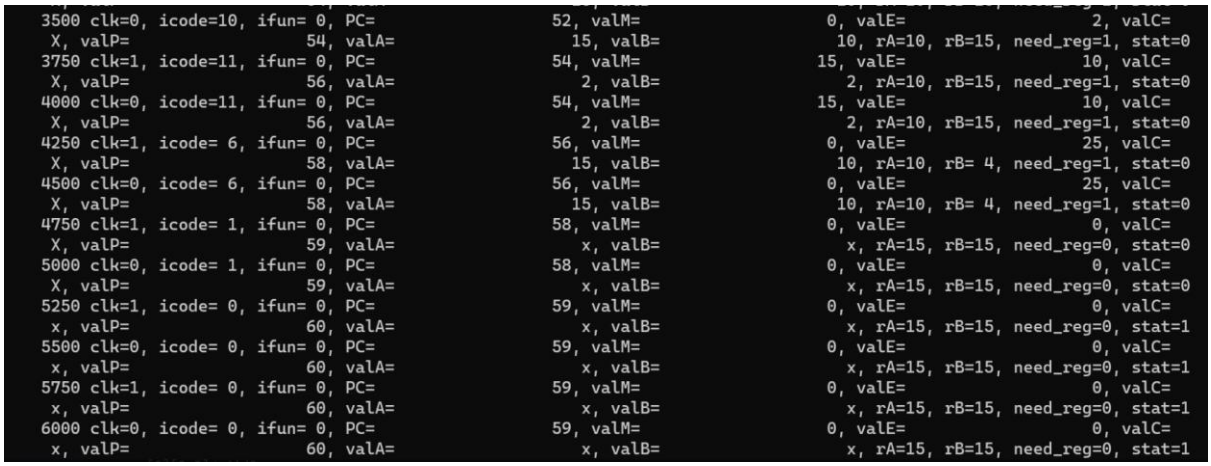
opq 10 4

nop

halt

Output:





Set 3:

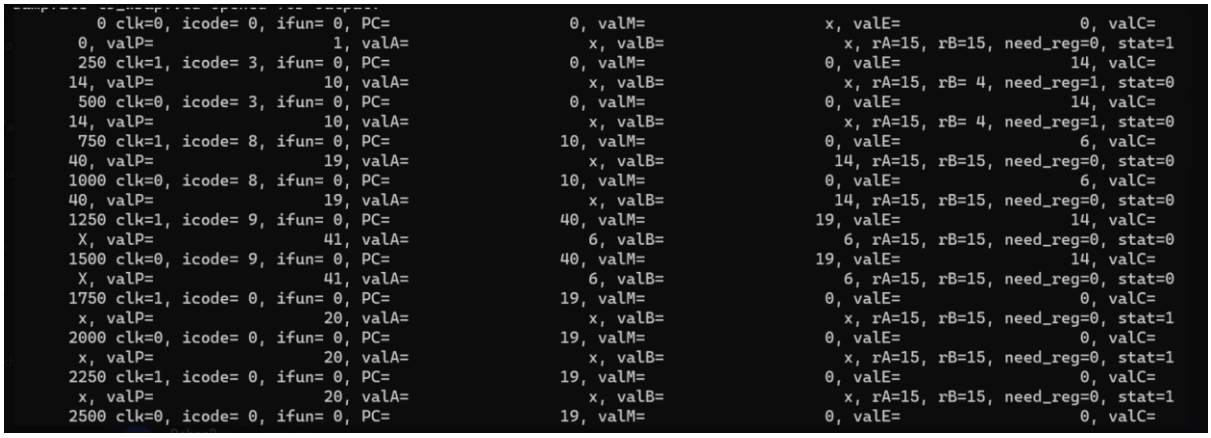
irmovq 14, 4

call 40

ret

halt

Output:



Set 4:

irmovq 14, 4

irmovq 6, 8

opq 6 1 4 8

cmovge 4 8

opq 6 0 4 8

jge 38

opq 6 0 4 8

halt 0 0

Output:

```
             0, valP=                  1, valA=                    x, valB=                    x, rA=15, rB=15, need_reg=0, stat=1
           250 clk=1, icode= 3, ifun= 0, PC=                   0, valM=                   0, valE=                   14, valC=
            14, valP=                 10, valA=                    x, valB=                    x, rA=15, rB= 4, need_reg=1, stat=0
           500 clk=0, icode= 3, ifun= 0, PC=                   0, valM=                   0, valE=                   14, valC=
            14, valP=                 10, valA=                    x, valB=                    x, rA=15, rB= 4, need_reg=1, stat=0
           750 clk=1, icode= 3, ifun= 0, PC=                  10, valM=                   0, valE=                    6, valC=
             6, valP=                 20, valA=                    x, valB=                    x, rA=15, rB= 8, need_reg=1, stat=0
          1000 clk=0, icode= 3, ifun= 0, PC=                  10, valM=                   0, valE=                    6, valC=
             6, valP=                 20, valA=                    x, valB=                    x, rA=15, rB= 8, need_reg=1, stat=0
          1250 clk=1, icode= 6, ifun= 1, PC=                  20, valM=                   0, valE=                   -8, valC=
42285167298597, valP=                 22, valA=                   14, valB=                    6, rA= 4, rB= 8, need_reg=1, stat=0
          1500 clk=0, icode= 6, ifun= 1, PC=                  20, valM=                   0, valE=                   -8, valC=
42285167298597, valP=                 22, valA=                   14, valB=                    6, rA= 4, rB= 8, need_reg=1, stat=0
          1750 clk=1, icode= 2, ifun= 5, PC=                  22, valM=                   0, valE=                   14, valC=
   6452204448, valP=                  24, valA=                   14, valB=                   -8, rA= 4, rB= 8, need_reg=1, stat=0
          2000 clk=0, icode= 2, ifun= 5, PC=                  22, valM=                   0, valE=                   14, valC=
   6452204448, valP=                  24, valA=                   14, valB=                   -8, rA= 4, rB= 8, need_reg=1, stat=0
          2250 clk=1, icode= 6, ifun= 0, PC=                  24, valM=                   0, valE=                   28, valC=
         9845, valP=                  26, valA=                   14, valB=                   14, rA= 4, rB= 8, need_reg=1, stat=0
          2500 clk=0, icode= 6, ifun= 0, PC=                  24, valM=                   0, valE=                   28, valC=
         9845, valP=                  26, valA=                   14, valB=                   14, rA= 4, rB= 8, need_reg=1, stat=0
          2750 clk=1, icode= 7, ifun= 5, PC=                  26, valM=                   0, valE=                    0, valC=
            38, valP=                 35, valA=                    x, valB=                    x, rA=15, rB=15, need_reg=0, stat=0
          3000 clk=0, icode= 7, ifun= 5, PC=                  26, valM=                   0, valE=                    0, valC=
            38, valP=                 35, valA=                    x, valB=                    x, rA=15, rB=15, need_reg=0, stat=0
          3250 clk=1, icode= 6, ifun= 0, PC=                  38, valM=                   0, valE=                   42, valC=
             X, valP=                 40, valA=                   14, valB=                   28, rA= 4, rB= 8, need_reg=1, stat=0
          3500 clk=0, icode= 6, ifun= 0, PC=                  38, valM=                   0, valE=                   42, valC=
             X, valP=                 40, valA=                   14, valB=                   28, rA= 4, rB= 8, need_reg=1, stat=0
          3750 clk=1, icode= 0, ifun= 0, PC=                  40, valM=                   0, valE=                    0, valC=
```