Intro to Processor Architecture

Y86-64 implementation in Verilog

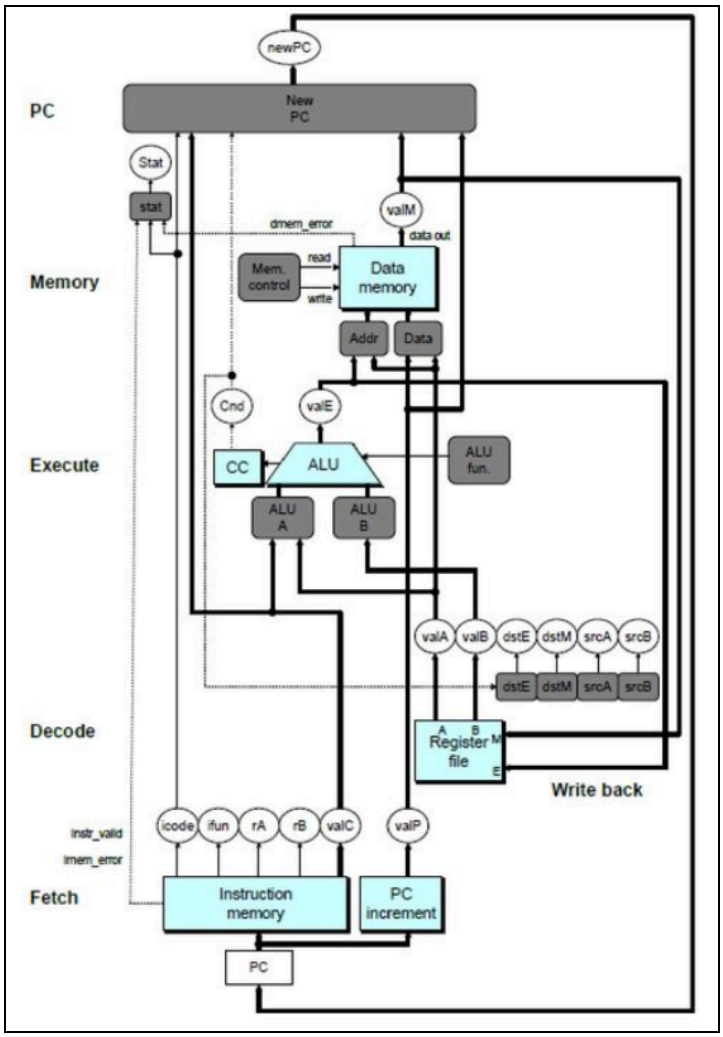
Team-34:

Druvitha E, 2022102029

Jyothi Sri P, 2022102056

SEQUENTIAL

- Fetch: Read instruction from instruction memory
- Decode: Read program registers
- Execute: Compute value or address
- Memory: Read or write data
- Write Back: Write program registers
- PC Update: Update program counter



Stage	HALT	NOP	CMOV	IRMOVQ
Fch	icode:ifun ← M_1[PC] valP ← PC + 1	icode:ifun ← M_1[PC] valP ← PC + 1	icode:ifun ← M_1[PC] rA:rB ← M_1[PC+1] valP ← PC + 2	icode:ifun ← M_1[PC] rA:rB ← M_1[PC+1] valC ← M_8[PC+2] valP ← PC + 10
Dec			valA ← R[rA]	
Exe	cpu.stat = HLT		valE ← valA Cnd ← Cond(CC,ifun)	valE ← valC
Mem				
WB			Cnd ? R[rB] ← valE	R[rB] ← valE
PC	PC ← 0	PC ← valP	PC ← valP	PC ← valP
Stage	**RMMOVQ**	**MRMOVQ**	**OPq**	**jXX**
Fch	icode:ifun ← M_1[PC] rA:rB ← M_1[PC+1] valC ← M_8[PC+2] valP ← PC + 10	icode:ifun ← M_1[PC] rA:rB ← M_1[PC+1] valC ← M_8[PC+2] valP ← PC + 10	icode:ifun ← M_1[PC] rA:rB ← M_1[PC+1] valP ← PC + 2	icode:ifun ← M_1[PC] valC ← M_8[PC+1] valP ← PC + 9
Dec	valA ← R[rA] valB ← R[rB]	 valB ← R[rB]	valA ← R[rA] valB ← R[rB]	
Exe	valE ← valB + valC	valE ← valB + valC	valE ← valB OP valA Set CC	Cnd ← Cond(CC,ifun)
Mem	M_8[valE] ← valA	valM ← M_8[valE]		
WB		R[rA] ← valM	R[rB] ← valE	
PC	PC ← valP	PC ← valP	PC ← valP	PC ← Cnd ? valC:valP
Stage	**CALL**	**RET**	**PUSHQ**	**POPQ**
Fch	icode:ifun ← M_1[PC] valC ← M_8[PC+1] valP ← PC + 9	icode:ifun ← M_1[PC] valP ← PC + 1	icode:ifun ← M_1[PC] rA:rB ← M_1[PC+1] valP ← PC + 2	icode:ifun ← M_1[PC] rA:rB ← M_1[PC+1] valP ← PC + 2
Dec	 valB ← R[RSP]	valA ← R[RSP] valB ← R[RSP]	valA ← R[rA] valB ← R[RSP]	valA ← R[RSP] valB ← R[RSP]
Exe	valE ← valB - 8	valE ← valB + 8	valE ← valB - 8	valE ← valB + 8
Mem	M_8[valE] ← valP	valM ← M_8[valA]	M_8[valE] ← valA	valM ← M_8[valA]
WB	R[RSP] ← valE	R[RSP] ← valE	R[RSP] ← valE	R[RSP] ← valE R[rA] ← valM
PC	PC ← valC	PC ← valM	PC ← valP	PC ← valP

FETCH

The fetch block takes in PC, instruction and returns icode, ifun, rA, rB, valC, valP.

```
if (PC > 1023)
    begin
        imem_error = 1'b1;
    end
else
    begin
        imem_error = 1'b0;
    end
```

If PC > 1023 (length of the instruction memory), this block gives instruction memory error (imem_error).

Else, if the PC is valid, based on the icode and ifun, this block assigns rA, rB, valC, and valP.

HALT

```
icode:ifun ← M1[PC]

valP ← PC + 1
```

NOP

```
icode:ifun ← M1[PC]

valP ← PC + 1
```

CMOV

```
icode:ifun ← M1[PC]
rA:rB ← M1[PC+1]

valP ← PC + 2
```

IRMOVQ

```
icode:ifun ← M1[PC]
rA:rB ← M1[PC+1]
valC ← M8[PC+2]
valP ← PC + 10
```

RMMOVQ

```
icode:ifun ← M1[PC]
rA:rB ← M1[PC+1]
valC ← M8[PC+2]
valP ← PC + 10
```

MRMOVQ

```
icode:ifun ← M1[PC]
rA:rB ← M1[PC+1]
valC ← M8[PC+2]
valP ← PC + 10
```

OPq

```
icode:ifun ← M1[PC]
rA:rB ← M1[PC+1]

valP ← PC + 2
```

jXX

```
icode:ifun ← M1[PC]

valC ← M8[PC+1]
valP ← PC + 9
```

CALL

```
icode:ifun ← M1[PC]

valC ← M8[PC+1]
valP ← PC + 9
```

RET

```
icode:ifun ← M1[PC]


valP ← PC + 1
```

PUSHQ

```
icode:ifun ← M1[PC]
rA:rB ← M1[PC+1]

valP ← PC + 2
```

POPQ

```
icode:ifun ← M1[PC]
rA:rB ← M1[PC+1]

valP ← PC + 2
```

1. **icode, ifun** defines the instructions to be performed.

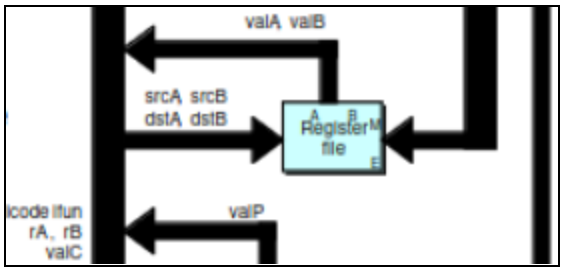
2. **rA, rB** specifies the registers on which these instructions are being performed.

3. **valC** is initialized with V/d/Dest according to the icode's.

4. **imem_error** occurs when icode or ifun are not in valid range. Rage of ifun is defined according to the current icode instruction.

5. **valP** is updated to current PC + instruction length.

DECODE



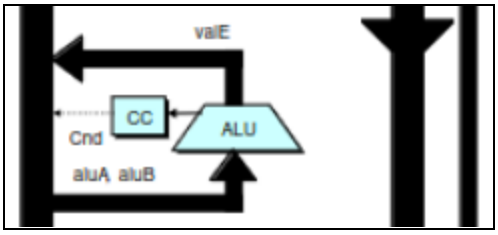
The decode block's function is to retrieve data from registers and allocate the values to valA and valB. To achieve this, a register file with 15 registers is required, and it undergoes updates during the program's execution.

We have initialized the registers with the following values

```
R[0] = 64'd9;
R[1] = 64'd8;
R[2] = 64'd7;
R[3] = 64'd6;
R[4] = 64'd9;
R[5] = 64'd4;
R[6] = 64'd3;
R[7] = 64'd2;
R[8] = 64'd1;
R[9] = 64'd10;
R[10] = 64'd12;
R[11] = 64'd13;
R[12] = 64'd14;
R[13] = 64'd15;
R[14] = 64'd16;
```

```
case (icode)
    // halt -> do nothing
    // nop -> do nothing
    4'b0010: begin   // cmovXX
        valA = R[rA];
        valB = 63'b0;
    end
    // irmovq -> do nothing
    4'b0100: begin   // rmmovq
        valA = R[rA];
        valB = R[rB];
    end
    4'b0101: begin   // mrmovq
        valB = R[rB];
    end
    4'b0110: begin   // Opq
        valA = R[rA];
        valB = R[rB];
    end
    // jXX -> do nothing
    4'b1000: begin   // call
        valB = R[4];
    end
    4'b1001: begin   // ret
        valA = R[4];
        valB = R[4];
    end
    4'b1010: begin   // pushq
        valA = R[rA];
        valB = R[4];
    end
    4'b1011: begin   // popq
        valA = R[4];
        valB = R[4];
    end
    default: ;
endcase
```

EXECUTE



The execution stage involves the computation of valE, which represents the effective address, using three Arithmetic Logic Units (ALUs). These ALUs take valA, valB, ifun, icode, and valC as inputs. Additionally, the execution stage is responsible for determining the condition codes (Cnd). The control of instruction execution is governed by the values of icode and ifun.

In the case of instructions like jXX and CMOV, condition codes are set, and valE is calculated through the ALU. The presence of CC_in and CC_out is a workaround due to Verilog limitations in modifying input values directly. For arithmetic operations, the value of CC_in is set to that of CC_out when condition codes are generated.

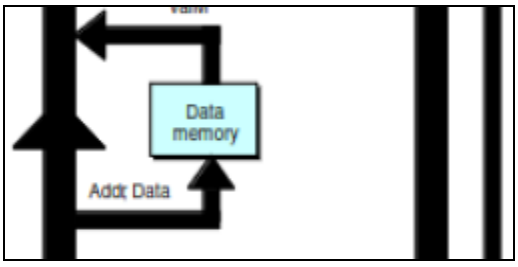
We update condition (cnd) only if the instruction is either CMOV or jXX based on the condition flags as shown below where ZF -> Zeroflag, SF-> Signedflag, OF-> Overflowflag

Effect	Set condition
$D \leftarrow$ ZF	Equal / zero
$D \leftarrow$ ~ZF	Not equal / not zero
$D \leftarrow$ SF	Negative
$D \leftarrow$ ~SF	Nonnegative
$D \leftarrow$ ~(SF ^ OF) & ~ZF	Greater (signed >)
$D \leftarrow$ ~(SF ^ OF)	Greater or equal (signed >=)
$D \leftarrow$ SF ^ OF	Less (signed <)
$D \leftarrow$ (SF ^ OF) | ZF	Less or equal (signed <=)

And then using ALU we execute valE according to their corresponding values of icode i.e for some cases it is valC+valB and in OPq case it is valB OPq valA and in other cases it is increment or decrement of Stack pointer.

```
always@(*) begin
    case(icode)
        4'b0010:
            valE = valA;
        4'b0011:
            valE = valC;
        4'b0100:
            valE = Movq_valE; // valC + valB
        4'b0101:
            valE = Movq_valE;
        4'b0110: begin
            valE = Opq_valE;  // valB OPq valA
            cndnflagout[0] <= (valE ==0) ? 1'b1:1'b0;
            cndnflagout[1] <= Opq_valE[63];
            cndnflagout[2] <= of2;
        end
        4'b1000:
            valE = Stkdec_valE;  // R[%rsp] - 8
        4'b1001:
            valE = Stkinc_valE;  // R[%rsp] + 8
        4'b1010:
            valE = Stkdec_valE;
        4'b1011:
            valE = Stkinc_valE;
    endcase
end
```

MEMORY



The memory stage is responsible for reading from and writing to memory. It obtains the memory address from the preceding stage and issues a request to the memory system for reading or writing data at that address. In case of a read request, the memory stage retrieves the data from memory and forwards it to the subsequent stage. If the request is a write, the memory stage stores the data in memory at the specified address.

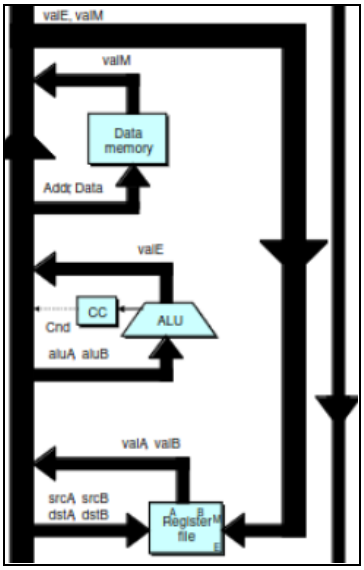
Furthermore, the memory stage is designed to detect memory-related errors, such as invalid memory accesses or page faults. It sets appropriate flags in the status register, which is then passed to the next stage. This status register can be utilized for decision-making or triggering interruptions.

A temporary memory has been implemented in the module to display transient changes in memory.

Based on the icode and ifun, following is changed/read in/from memory:

```
case (icode)
    4'b0100: begin
        Mo[valE] = valA;  // rmmovq
        // $display("icode: %4b Mem allocated %d at valE", icode,valA, valE);
    end
    4'b0101: begin
        valM = Mo[valE];  // mrmovq
        // $display("icode: %4b Value read %d from valE %d",icode,valM, valE);
    end
    4'b1000: begin
        Mo[valE] = valP;  // call
        // $display("icode: %4b Mem allocated %d at valE", icode,valP, valE);
    end
    4'b1001: begin
        valM = Mo[valA];  // ret
        // $display("icode: %4b Value read %d",icode,valM);
    end
    4'b1010: begin
        Mo[valE] = valA;  // pushq
        // $display("icode: %4b Mem allocated %d at valE", icode,valA, valE);
    end
    4'b1011: begin
        valM = Mo[valA];  // popq
        // $display("icode: %4b Value read %d",icode,valM);
    end
    default: ;
endcase
```

WRITEBACK

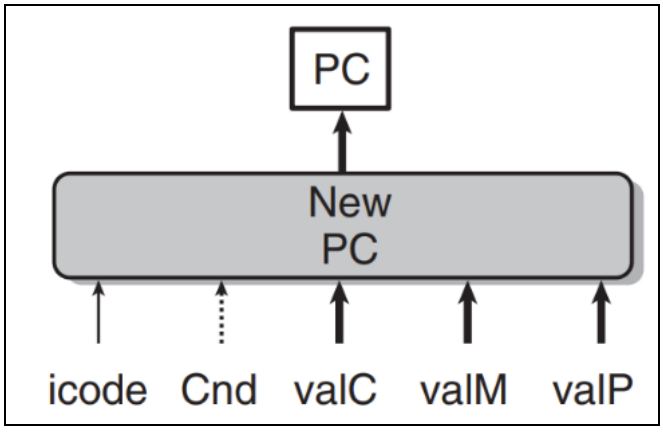


This stage is responsible for updating the register file based on the icode and ifun of the current instruction. The final values of valE and valM are passed to the register file, and the register is updated accordingly. The specific update operation depends on the value of icode, ensuring that either valE or valM contributes to the final register value.

Based on icode and ifun, following is changed (written back) in registers:

```
case(icode)
    4'b0010:
        begin
            if (cnd)
                Regout[rB] = valE; // cmov
        end
    4'b0011:
        begin
            Regout[rB] = valE; // irmovq
        end
    4'b0101:
        begin
            Regout[rA] = valM; // mrmovq
        end
    4'b0110:
        begin
            Regout[rB] = valE; // opq
        end
    4'b1000:
        begin
          Regout[4] = valE; // call
        end
    4'b1001:
        begin
          Regout[4] = valE; // ret
        end
    4'b1010:
        begin
          Regout[4] = valE; // pushq
        end
    4'b1011:
        begin
            Regout[4] = valE;
            Regout[rA] = valM;  // popq
        end

endcase
```

PC_UPDATE



The purpose of the PC update is to point to the address of the next instruction. The PC Update block has icode, Cnd, valC, valM and valP as inputs and the value of updated PC as output. The PC Update part works on switch statements which selects the instructions according to the values of icode and Cnd and updates the value of PC accordingly.

```
case (icode)
    4'b0000: new_pc = 63'b0;   // halt
    4'b0001: new_pc = valP;    // nop
    4'b0010: new_pc = valP;    // cmovXX
    4'b0011: new_pc = valP;    // irmovq
    4'b0100: new_pc = valP;    // rmmovq
    4'b0101: new_pc = valP;    // mrmovq
    4'b0110: new_pc = valP;    // Opq
    4'b0111: begin             // jXX
        if(cnd == 1)
            new_pc = valC;
        else
            new_pc = valP;
    end
    4'b1000: new_pc = valC;    // call
    4'b1001: new_pc = valM;    // ret
    4'b1010: new_pc = valP;    // pushq
    4'b1011: new_pc = valP;    // popq
    default: new_pc = valP;
endcase
```

RESULTS

1)

```
irmovq $0x100, %rbx
irmovq $0x200, %rdx
addq %rdx, %rbx
```

```
15ns   FETCH ---> icode:ifun =  3: 0 rA=15 rB= 3,valC=               256
       DECODE --> valA:                  x;  valB:                 x
       EXECUTE--> valE:               256, cndn = 0, cndnflagout = xxx
       MEMORY --> valM:                  x
       WRITEBACK--> R0:    9, R1:    8, R2:    7, R3:  256, R4:    9, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
       PC UPD --> PC:                 10
```

```
45ns   FETCH ---> icode:ifun =  3: 0 rA=15 rB= 2,valC=               512
       DECODE --> valA:                  x;  valB:                 x
       EXECUTE--> valE:               512, cndn = 0, cndnflagout = xxx
       MEMORY --> valM:                  x
       WRITEBACK--> R0:    9, R1:    8, R2:  512, R3:  256, R4:    9, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
       PC UPD --> PC:                 20
```

```
75ns   FETCH ---> icode:ifun =  6: 0 rA= 2 rB= 3,valC=               512
       DECODE --> valA:                512;  valB:               256
       EXECUTE--> valE:                768, cndn = 0, cndnflagout = 000
       MEMORY --> valM:                  x
       WRITEBACK--> R0:    9, R1:    8, R2:  512, R3:  768, R4:    9, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
       PC UPD --> PC:                 22
```

2)

```
irmovq $0x100, %rbx
rmmovq %rbx, 3(%rax)
mrmovq %rcx, 3(%rax)
```

```
15ns   FETCH ---> icode:ifun =  3: 0 rA=15 rB= 3,valC=               256
       DECODE --> valA:                  x;  valB:                   x
       EXECUTE--> valE:                256, cndn = 0, cndnflagout = xxx
       MEMORY --> valM:                  x
       WRITEBACK--> R0:    9, R1:    8, R2:    7, R3:  256, R4:    9, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
       PC UPD --> PC:                 10
```

```
45ns   FETCH ---> icode:ifun =  4: 0 rA= 3 rB= 0,valC=                 3
       DECODE --> valA:                256;  valB:                   9
       EXECUTE--> valE:                 12, cndn = 0, cndnflagout = xxx
       MEMORY --> valM:                  x
       WRITEBACK--> R0:    9, R1:    8, R2:    7, R3:  256, R4:    9, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
       PC UPD --> PC:                 20
```

```
75ns   FETCH ---> icode:ifun =  5: 0 rA= 1 rB= 0,valC=                 3
       DECODE --> valA:                256;  valB:                   9
       EXECUTE--> valE:                 12, cndn = 0, cndnflagout = xxx
       MEMORY --> valM:                256
       WRITEBACK--> R0:    9, R1:  256, R2:    7, R3:  256, R4:    9, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
       PC UPD --> PC:                 30
```

3)

```
irmovq $4, %rax
irmovq $5, %rbx
subq %rax, %rbx
jg L2
irmovq $1, %r8
jmp end
L2: irmovq $2, %r8
end:
```

```
15ns   FETCH ---> icode:ifun =  3: 0 rA=15 rB= 0,valC=                 4
       DECODE --> valA:                  x;  valB:                   x
       EXECUTE--> valE:                  4, cndn = 0, cndnflagout = xxx
       MEMORY --> valM:                  x
       WRITEBACK--> R0:    4, R1:    8, R2:    7, R3:    6, R4:    9, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
       PC UPD --> PC:                 10
```

```
45ns   FETCH ---> icode:ifun =  3: 0 rA=15 rB= 3,valC=                 5
       DECODE --> valA:                  x;  valB:                   x
       EXECUTE--> valE:                  5, cndn = 0, cndnflagout = xxx
       MEMORY --> valM:                  x
       WRITEBACK--> R0:    4, R1:    8, R2:    7, R3:    5, R4:    9, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
       PC UPD --> PC:                 20
```

```
75ns   FETCH ---> icode:ifun =  6: 1 rA= 0 rB= 3,valC=                5
       DECODE --> valA:                  4;  valB:                  5
       EXECUTE--> valE:                  1, cndn = 0, cndnflagout = 000
       MEMORY --> valM:                  x
       WRITEBACK--> R0:    4, R1:    8, R2:    7, R3:    1, R4:    9, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
       PC UPD --> PC:                 22
```

```
105ns   FETCH ---> icode:ifun =  7: 6 rA= 0 rB= 3,valC=               50
       DECODE --> valA:                  4;  valB:                  5
       EXECUTE--> valE:                  1, cndn = 1, cndnflagout = 000
       MEMORY --> valM:                  x
       WRITEBACK--> R0:    4, R1:    8, R2:    7, R3:    1, R4:    9, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
       PC UPD --> PC:                 50
```

```
135ns   FETCH ---> icode:ifun =  3: 0 rA=15 rB= 8,valC=                2
       DECODE --> valA:                  4;  valB:                  5
       EXECUTE--> valE:                  2, cndn = 0, cndnflagout = 000
       MEMORY --> valM:                  x
       WRITEBACK--> R0:    4, R1:    8, R2:    7, R3:    1, R4:    9, R5:    4, R6:    3, R7:    2, R8:    2, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
       PC UPD --> PC:                 60
```

4)

```
call 29
irmovq 2^6, %rcx
irmovq 2^6, %rax
irmovq $0x100, %rcx
irmovq $0x100, %rax
ret
```

```
 15ns   FETCH ---> icode:ifun =  8: 0 rA= x rB= x,valC=               29
       DECODE --> valA:                  x;  valB:                  9
       EXECUTE--> valE:                  1, cndn = 0, cndnflagout = xxx
       MEMORY --> valM:                  x
       WRITEBACK--> R0:    9, R1:    8, R2:    7, R3:    6, R4:    1, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
       PC UPD --> PC:                 29
```

```
 45ns   FETCH ---> icode:ifun =  3: 0 rA=15 rB= 1,valC=              256
       DECODE --> valA:                  x;  valB:                  9
       EXECUTE--> valE:                256, cndn = 0, cndnflagout = xxx
       MEMORY --> valM:                  x
       WRITEBACK--> R0:    9, R1:  256, R2:    7, R3:    6, R4:    1, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
       PC UPD --> PC:                 39
```

```
 75ns   FETCH ---> icode:ifun =  3: 0 rA=15 rB= 0,valC=              256
       DECODE --> valA:                  x;  valB:                  9
       EXECUTE--> valE:                256, cndn = 0, cndnflagout = xxx
       MEMORY --> valM:                  x
       WRITEBACK--> R0:  256, R1:  256, R2:    7, R3:    6, R4:    1, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
       PC UPD --> PC:                 49
```

```
105ns   FETCH ---> icode:ifun =  9: 0 rA=15 rB= 0,valC=              256
       DECODE --> valA:                  1;  valB:                  1
       EXECUTE--> valE:                  9, cndn = 0, cndnflagout = xxx
       MEMORY --> valM:                  9
       WRITEBACK--> R0:  256, R1:  256, R2:    7, R3:    6, R4:    9, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
       PC UPD --> PC:                  9
```

```
135ns   FETCH ---> icode:ifun =  3: 0 rA=15 rB= 1,valC=                64
        DECODE --> valA:                  1;  valB:                  1
        EXECUTE--> valE:                 64, cndn = 0, cndnflagout = xxx
        MEMORY --> valM:                  9
        WRITEBACK--> R0:  256, R1:   64, R2:    7, R3:    6, R4:    9, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
        PC UPD --> PC:                   19
```

```
165ns   FETCH ---> icode:ifun =  3: 0 rA=15 rB= 0,valC=                64
        DECODE --> valA:                  1;  valB:                  1
        EXECUTE--> valE:                 64, cndn = 0, cndnflagout = xxx
        MEMORY --> valM:                  9
        WRITEBACK--> R0:   64, R1:   64, R2:    7, R3:    6, R4:    9, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
        PC UPD --> PC:                   29
```

```
195ns   FETCH ---> icode:ifun =  3: 0 rA=15 rB= 1,valC=                256
        DECODE --> valA:                  1;  valB:                  1
        EXECUTE--> valE:                 256, cndn = 0, cndnflagout = xxx
        MEMORY --> valM:                  9
        WRITEBACK--> R0:   64, R1:  256, R2:    7, R3:    6, R4:    9, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
        PC UPD --> PC:                   39
```

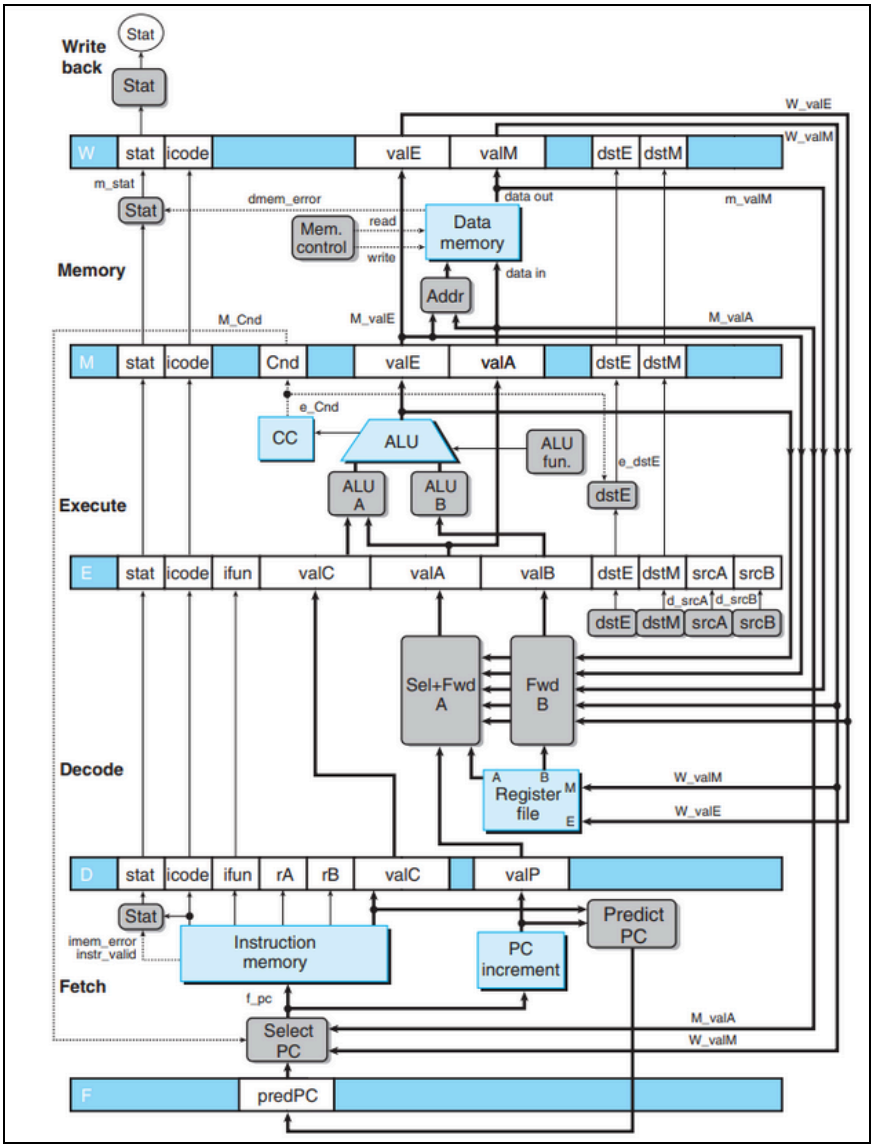
```
225ns   FETCH ---> icode:ifun =  3: 0 rA=15 rB= 0,valC=                256
        DECODE --> valA:                  1;  valB:                  1
        EXECUTE--> valE:                 256, cndn = 0, cndnflagout = xxx
        MEMORY --> valM:                  9
        WRITEBACK--> R0:  256, R1:  256, R2:    7, R3:    6, R4:    9, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
        PC UPD --> PC:                   49
```

```
255ns   FETCH ---> icode:ifun =  9: 0 rA=15 rB= 0,valC=                256
        DECODE --> valA:                  9;  valB:                  9
        EXECUTE--> valE:                 17, cndn = 0, cndnflagout = xxx
        MEMORY --> valM:                  x
        WRITEBACK--> R0:  256, R1:  256, R2:    7, R3:    6, R4:   17, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
        PC UPD --> PC:                   x
```

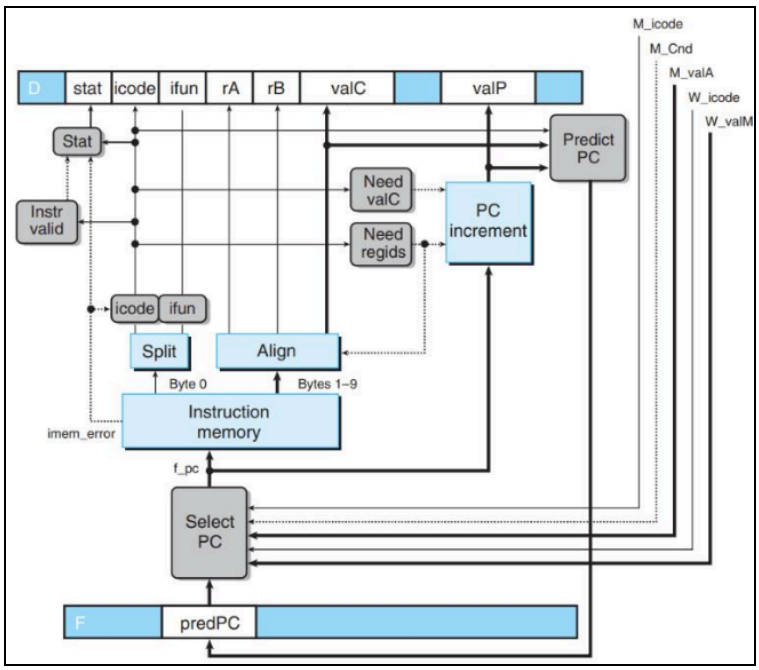
```
285ns   FETCH ---> icode:ifun =  x: x rA=15 rB= 0,valC=                256
        DECODE --> valA:                  9;  valB:                  9
        EXECUTE--> valE:                 17, cndn = 0, cndnflagout = xxx
        MEMORY --> valM:                  x
        WRITEBACK--> R0:  256, R1:  256, R2:    7, R3:    6, R4:   17, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
        PC UPD --> PC:                   50
```

PIPELINE

The pipeline implementation required the introduction of registers between each stage, allowing specific outputs from one stage to serve as inputs for the next. An alteration was made in the PC update process; instead of having a distinct PC update block, it was integrated with the Fetch stage. This modification ensures that before each instruction executes, the Fetch stage determines both the current PC value and the next predicted PC. Additionally, control logic was incorporated to handle Bubble, Stall, and Forwarding, aiming to address pipeline hazards.



FETCH



This module takes in an instruction and the PC to determine the icode, ifun, rA, rB, valC, valP, predicted pc (for fetch of next instruction in next clock cycle), in the positive edge of the clk cycle. In the negative edge of the clk cycle it updates the D register based on stalls and bubbles.

The module also handles various scenarios, including halting conditions, address errors, and invalid instructions. Signals such as F_stall, D_stall, and D_bubble influence the control flow, allowing synchronization with the pipeline stages.

Following are its arguments:

```
module fetch_64(input [63:0]W_valM, input [3:0]M_icode, input
[3:0]W_icode, input M_cnd, input [63:0] M_valA, input [63:0] Jxx_Pred,
input Jxx_cnd, input F_stall, input D_bubble, input D_stall, input [0:79]
instruction, input [63:0] F_PC , input clk, output reg [3:0] f_icode,
output reg [3:0] f_ifun, output reg [3:0] f_rA, output reg [3:0] f_rB,
output reg [63:0] f_valC, output reg [63:0] f_valP, output reg [3:0]
f_stat,  output reg [63:0] f_PC, output reg [63:0] Temp_predPC, input fi);
```

PC PREDICTION STRATEGY:

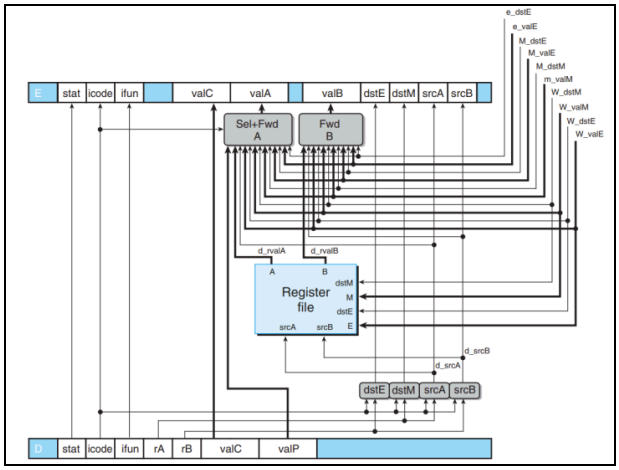
Instructions that Don't Transfer Control
- Predict next PC to be valP
- Always reliable

Call and Unconditional Jumps
- Predict next PC to be valC (destination)
- Always reliable

Conditional Jumps
- Predict next PC to be valC (destination)
- Only correct if branch is taken
  - Typically right 60% of time

Return Instruction
- Don't try to predict

DECODE



This module, in the positive edge of the clk cycle, updates d_srcA, d_srcB, d_dstE, d_dstM, (useful for data forwarding). In negative edge, it implements the blocks Sel+Fwd A, Fwd B. Based on if there's a bubble in execute stage, it updates the 'E' pipeline register.

```
module decode_64 (input clk, input E_bubble, input [3:0] D_stat, input [3:0] D_icode, input
[3:0] D_ifun, input [3:0] D_rA, input [3:0] D_rB, input[63:0] D_valC ,input[63:0] D_valP,
input [63:0] [0:14] R, output reg [3:0] E_stat, output reg [3:0] E_icode, output reg [3:0]
E_ifun, output reg [63:0] E_valC, output reg [63:0] E_valA, output reg [63:0] E_valB, output
reg [3:0] E_dstE, output reg [3:0] E_dstM, output reg [3:0] E_srcA, output reg [3:0] E_srcB,
input [63:0] W_valE, input [63:0] W_valM, input [63:0] m_valM, input [63:0] M_valA, input
[63:0] M_valE, input [63:0] e_valE, output reg [3:0] d_srcA, output reg [3:0] d_srcB, input
[3:0] e_dstE, input [3:0] M_dstM, input [3:0] M_dstE, input [3:0] W_dstM, input [3:0] W_dstE);
```

Setting d_ values:

```
// halt -> do nothing

// nop -> do nothing

// cmovxx
if(D_icode == 4'b0010)
begin
    d_origvalA=R[D_rA];
    d_origvalB = 0;
    d_srcA = D_rA;
    d_dstE = D_rB;
end

//irmovq
else if(D_icode == 4'b0011)
begin
    d_dstE = D_rB;
end

//rmmovq
else if(D_icode == 4'b0100)
begin
    d_origvalA = R[D_rA];
    d_origvalB = R[D_rB];
    d_srcA = D_rA;
    d_srcB = D_rB;
end

//mrmovq
else if (D_icode == 4'b0101)
begin
    d_origvalB = R[D_rB];
    d_srcB = D_rB;
    d_dstM = D_rA;
end

//opq
else if(D_icode == 4'b0110)
begin
    d_origvalA = R[D_rA];
    d_origvalB = R[D_rB];
    d_srcA = D_rA;
    d_srcB = D_rB;
    d_dstE = D_rB;
end
```

```
//jxx -> do nothing

//call
else if(D_icode == 4'b1000)
begin
    d_origvalB = R[4];
    d_srcB = 4;
    d_dstE = 4;
end

// ret
else if(D_icode == 4'b1001)
begin
    d_origvalA = R[4];
    d_origvalB = R[4];
    d_srcA = 4;
    d_srcB = 4;
    d_dstE = 4;
end

//push
else if (D_icode == 4'b1010)
begin
    d_origvalA = R[D_rA];
    d_origvalB = R[4];
    d_srcA = D_rA;
    d_srcB = 4;
    d_dstE = 4;
end

//pop
else if(D_icode == 4'b1011)
begin
    d_origvalA = R[4];
    d_origvalB = R[4];
    d_srcA = 4;
    d_srcB = 4;
    d_dstE = 4;
    d_dstM = D_rA;
end
```

Sel + Fwd A block:

```
// Sel+Fwd A (order -> execute, memory, write back)
if(D_icode==4'b0111 | D_icode == 4'b1000) //jxx or call
    d_valA = D_valP;
else if(e_dstE != 4'b1111 && d_srcA == e_dstE)
begin
    $display("DATA FORWARDING");
    d_valA = e_valE;
end
else if(M_dstM != 4'b1111 & d_srcA == M_dstM)
begin
    d_valA = m_valM;
    $display("DATA FORWARDING");
end
else if(W_dstM != 4'b1111 & d_srcA == W_dstM)
begin
    d_valA = W_valM;
    $display("DATA FORWARDING");
end
else if(M_dstE != 4'b1111 & d_srcA == M_dstE)
begin
    $display("DATA FORWARDING");
    d_valA = M_valE;
end
else if(W_dstE!=4'b1111 & d_srcA==W_dstE)
begin
    d_valA = W_valE;
    $display("DATA FORWARDING");
end
else begin
    // $display("DATA FORWA----------d_srcA = %b; e_dstE = %b
    d_valA = d_origvalA;
end
```

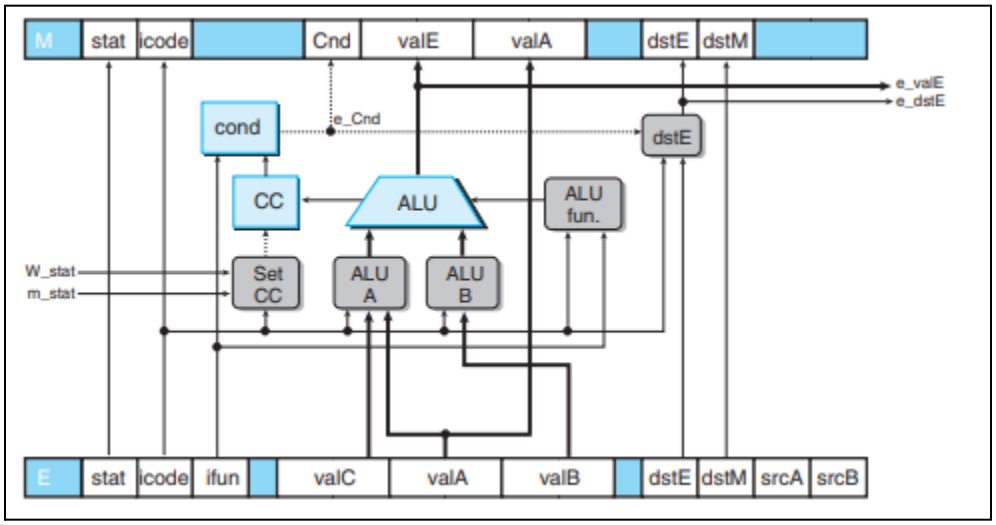
Fwd B block:

```
// Fwd B
if(e_dstE != 4'hF & d_srcB == e_dstE)
    d_valB = e_valE;
else if(M_dstM != 4'hF & d_srcB == M_dstM)
    d_valB = m_valM;
else if(W_dstM != 4'hF & d_srcB == W_dstM)
    d_valB = W_valM;
else if(M_dstE != 4'hF & d_srcB == M_dstE)
    d_valB = M_valE;
else if(W_dstE != 4'hF & d_srcB == W_dstE)
    d_valB = W_valE;
else
    d_valB = d_origvalB;
```

Below is the forwarding logic

```
## What should be the A value?
int d_valA = [
  # Use incremented PC
     D_icode in { ICALL, IJXX } : D_valP;
  # Forward valE from execute
     d_srcA == e_dstE : e_valE;
  # Forward valM from memory
     d_srcA == M_dstM : m_valM;
  # Forward valE from memory
     d_srcA == M_dstE : M_valE;
  # Forward valM from write back d_srcA ==
W_dstM : W_valM;
  # Forward valE from write back
     d_srcA == W_dstE : W_valE;
  # Use value read from register file
     1 : d_rvalA;
];
```

EXECUTE



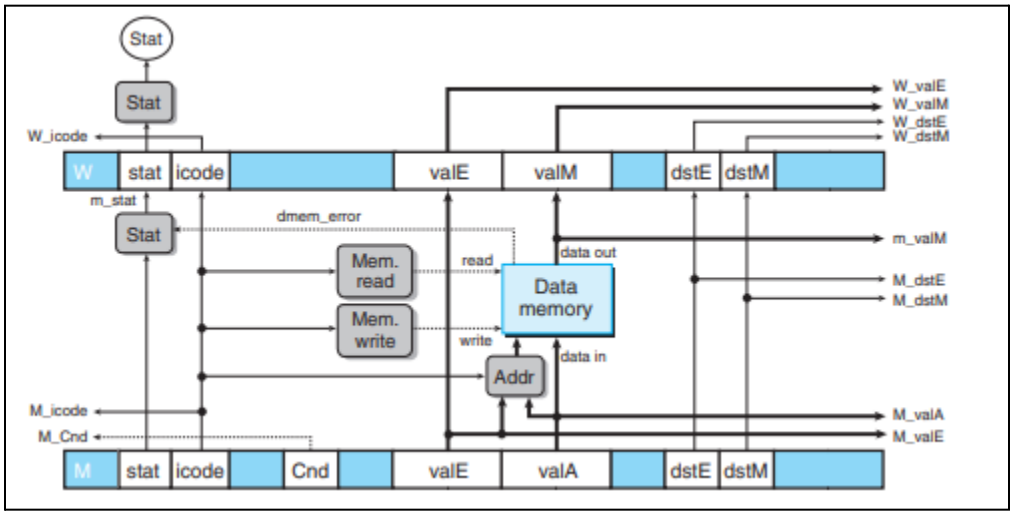
Above figure shows the way of implementation of the Execute stage in the pipeline. The hardware units and logic blocks are the same as those in SEQ, with signals renamed accordingly. Notably, signals like e_valE and e_dstE are routed to the decode stage for forwarding. However, there is a variation in the logic responsible for determining whether to update condition codes, referred to as "Set CC." This logic incorporates inputs m_stat and W_stat to identify instances where an instruction causing an exception is progressing through later pipeline stages. In such cases, updating of condition codes is inhibited.

Here we are setting cnd according to the condition flags when E_icode is either 4'b0010 (Cmov) or 4'b0111 (Jxx). In Execute stage we update e_dstE depending on this condition i.e if cnd = 1 we update it otherwise we set e_dstE to 15.

Execute stage in Pipeline is similar to execute stage in Sequential other than forwarding e_dstE and e_valE to decode stage. And updating them in negedge clk.

```
always@(negedge clk) begin
    e_stat <= stat;
    e_icode <= icode;
    e_dstM <= dstM;
    e_valA <= valA;
    // $display("Entered here\n");
    M_dstE <= e_dstE;
    M_valE <= valE;
end
```

MEMORY



Above figure shows the way of implementation of the Memory stage in the pipeline. Many of the signals from pipeline registers M(Memory) and W(Writeback) are passed down to earlier stages to provide write-back results, instruction addresses, and forwarded results.

Comparing PIPE's memory stage to SEQ's, it's observed that the "Mem. data" block in SEQ, responsible for choosing between data sources valP and valA, is absent. Instead, this selection function is now integrated into the "Sel+Fwd A" block within the decode stage. Other blocks in this stage remain largely unchanged from SEQ, with signal names adjusted accordingly. Additionally, various values stored in pipeline registers, M, and W are provided to other circuit components as part of forwarding and pipeline control logic.

```
always@(posedge clk)
begin
    for (i=0; i<1024; i=i+1) begin
        Mout[i] = Min[i];
    end

    case (M_icode)
        4'b0000: ;      // halt -> do nothing
        4'b0001: ;      // nop -> do nothing
        4'b0010: ;      // cmov -> do nothing
        4'b0011: ;      // irmov -> do nothing
        4'b0100: begin      // rmmov
            if (M_valE > 1023)
                dmem_error = 1;
            else
                Mout[M_valE] = M_valA;
        end
        4'b0101: begin      // mrmov
            if (M_valE > 1023)
                dmem_error = 1;
            else
```

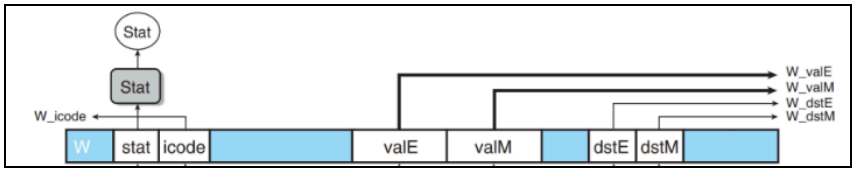
```
                m_valM = Mout[M_valE];
        end
        4'b0110: ;      // opq -> do nothing
        4'b0111: ;      // jxx -> do nothing
        4'b1000: begin      // call
            if (M_valE > 1023)
                dmem_error = 1;
            else
                Mout[M_valE] = M_valA;
        end
        4'b1001: begin      // ret
            if (M_valA > 1023)
                dmem_error = 1;
            else
                m_valM = Mout[M_valA];
        end
        4'b1010: begin      // push
            if (M_valE > 1023)
                dmem_error = 1;
            else
                Mout[M_valE] = M_valA;
        end
        4'b1011: begin      // pop
            if (M_valA > 1023)
                dmem_error = 1;
            else
```

dmem_Error and either reading/updating from/to memory are the main things that happen in memory block

Arguments of the block:

```
module memory_64 (input clk, input [63:0] [0:1023] Min, output reg [63:0]
[0:1023] Mout, input [3:0] M_stat, input [3:0] M_icode, input M_cnd, input
[63:0] M_valE, input [63:0] M_valA, input [3:0] M_dstE, input [3:0]
M_dstM, output reg [3:0] W_stat, output reg [3:0] W_icode, output reg
[63:0] W_valE, output reg [63:0] W_valM, output reg [3:0] W_dstE, output
reg [3:0] W_dstM, output reg [63:0] m_valM, output reg [3:0] m_stat);
```

WRITE BACK



This module takes in W_dstE/W_dstM and updates this register to W_valE/W_valM.

```
case (W_icode)
    4'b0000: ;      // halt -> do nothing
    4'b0001: ;      // nop -> do nothing
    4'b0010: begin      // cmov
        if (cnd)
            Regout[W_dstE] = W_valE;
    end
    4'b0011: begin      // irmov
        Regout[W_dstE] = W_valE;
    end
    4'b0100: ;      // rmmov -> don nothing
    4'b0101: begin      // mrmov
        Regout[W_dstM] = W_valM;
    end
    4'b0110: begin      // opq
        $display("YOYO %b %b",W_dstE, W_valE);
        Regout[W_dstE] = W_valE;
    end
    4'b0111: ;      // jxx -> do nothing
    4'b1000: begin      // call
        Regout[W_dstE] = W_valE;
    end
```

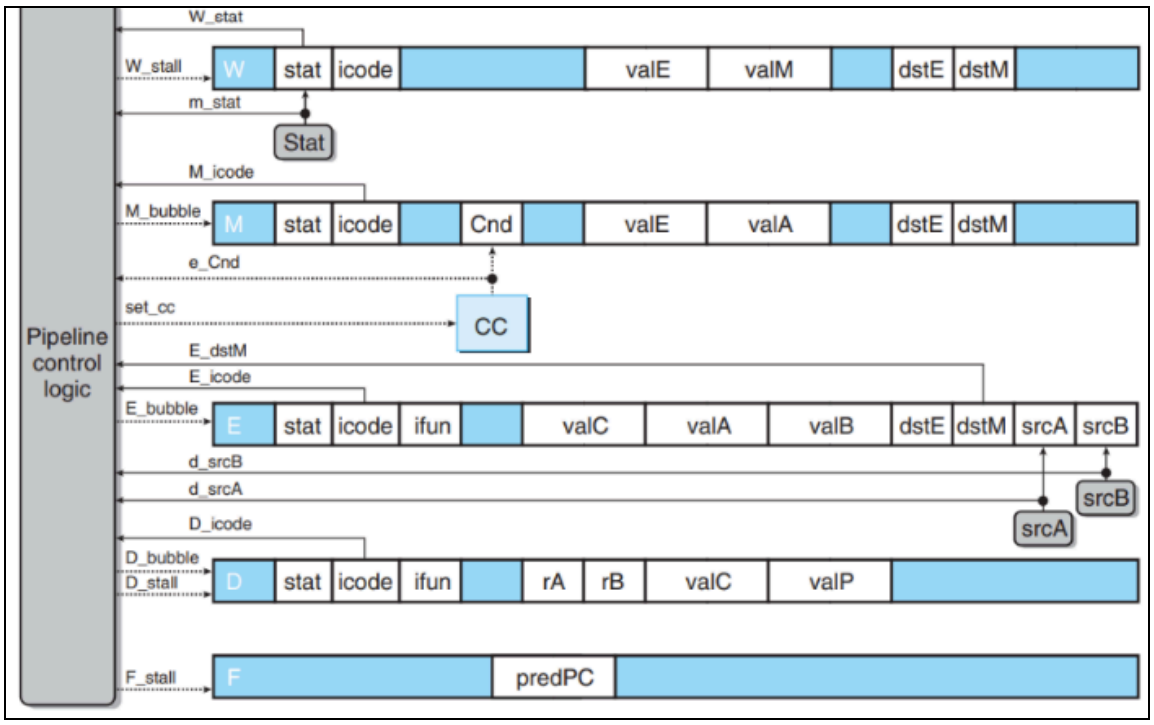
```
    4'b1001: begin      // ret
        Regout[W_dstE] = W_valE;
    end
    4'b1010: begin      // push
        Regout[W_dstE] = W_valE;
    end
    4'b1011: begin      // pop
        Regout[W_dstE] = W_valE;
        Regout[W_dstM] = W_valM;
    end
endcase
```

Arguments of the block:

```
module writeback_64(input clk, input [3:0] W_stat, input [3:0] W_icode,
input [63:0] W_valE, input [63:0] W_valM, input [3:0] W_dstE, input [3:0]
W_dstM, input [63:0] [0:14] Regin , output reg [63:0] [0:14] Regout, input
cnd);
```

CONTROL LOGIC

(Bubble,Stall and Forwarding)



Detection of these data hazards:

Condition	Trigger
Processing ret	IRET in { D_icode, E_icode, M_icode }
Load/Use Hazard	E_icode in { IMRMOVQ, IPOPQ } && E_dstM in { d_srcA, d_srcB }
Mispredicted Branch	E_icode = IJXX & !e_Cnd

The conditions for implementing these for few data hazards are as follows:

Condition	F	D	E	M	W
Processing ret	stall	bubble	normal	normal	normal
Load/Use Hazard	stall	stall	bubble	normal	normal
Mispredicted Branch	normal	bubble	bubble	normal	normal

Based on the above, below is the piece of code in the integrated module to trigger control logic:

```
if(E_icode == 4'b0000 | m_stat!=4'b1000 | W_stat!=4'b1000)
    begin
        E_cndnflagin = 0;
        F_stall = 0;
        D_stall = 0;
        E_bubble = 0;
        D_bubble = 0;
    end
else if(E_icode == 4'b0111 & !M_cnd) begin
    F_stall = 0;
    D_stall = 0;
    D_bubble = 1;
    E_bubble = 1;
end
else if ((E_icode == 4'b0101 | E_icode == 4'b1011) & (M_dstM==d_srcA | M_dstM==d_srcB)) begin
    F_stall = 1;
    D_stall = 1;
    E_bubble = 1;
    D_bubble = 0;
 end
 else if (E_icode == 4'b1001 | M_icode == 4'b1001 | D_icode == 4'b1001) begin
        F_stall = 1;
        D_bubble = 1;
        D_stall = 0;
        E_bubble = 0;
 end
 else begin
        F_stall = 0;
        D_stall = 0;
        E_bubble = 0;
        D_bubble = 0;
 end
```

Error handling

Status codes

1000 - INS

0100 - ADR

0010 - HLT

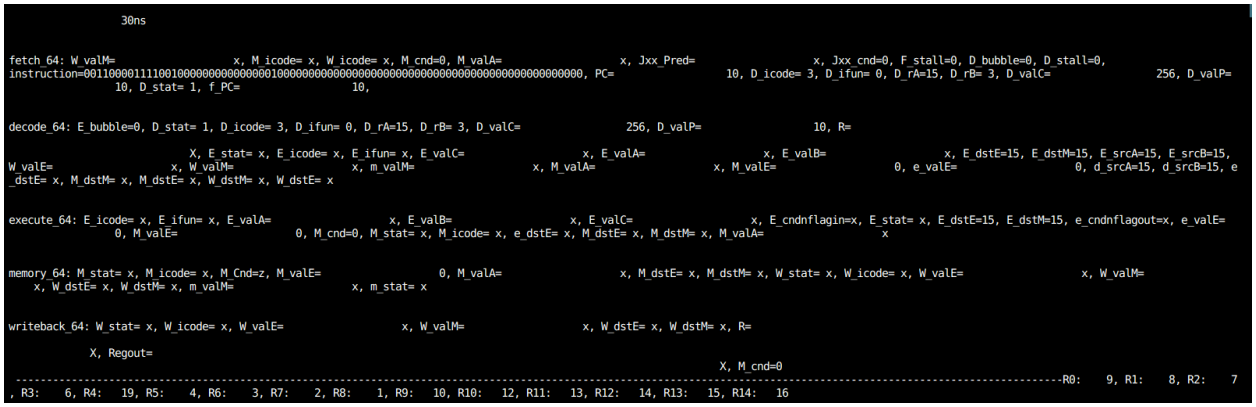
0001 - AOK

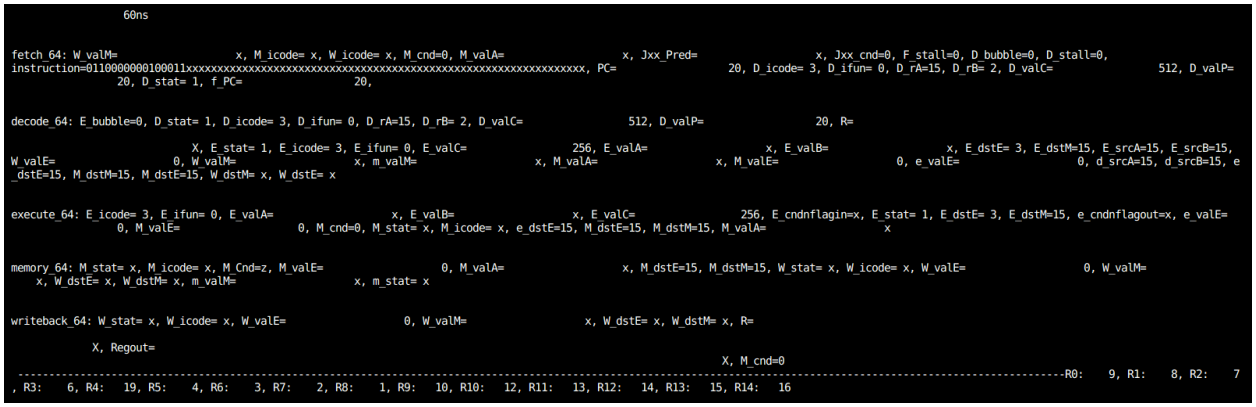
```
always@(*) begin
    if ( W_stat[1] == 1'b1 ) begin
      $display("HALT Encountered \n");
        $finish;
    end
    if (W_stat[2] == 1'b1 ) begin
        $display("Wrong Address Encountered \n");
        $finish;
    end
    if (  W_stat[3] == 1'b1 ) begin
        $display("Wrong Instruction Encountered \n");
        $finish;
    end
end
```

RESULTS

1)

```
irmovq $0x100, %rbx
irmovq $0x200, %rdx
addq %rdx, %rbx
```





```
                90ns

fetch_64: W_valM=                   x, M_icode= 3, W_icode= x, M_cnd=0, M_valA=                   x, Jxx_Pred=                   x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, PC=                  22, D_icode= 6, D_ifun= 0, D_rA= 2, D_rB= 3, D_valC=                 512, D_valP=
                 22, D_stat= 1, f_PC=                   22,

decode_64: E_bubble=0, D_stat= 1, D_icode= 6, D_ifun= 0, D_rA= 2, D_rB= 3, D_valC=                 512, D_valP=                  22, R=

                             X, E_stat= 1, E_icode= 3, E_ifun= 0, E_valC=                 512, E_valA=                   x, E_valB=                   x, E_dstE= 2, E_dstM=15, E_srcA=15, E_srcB=15,
W_valE=                    0, W_valM=                   x, m_valM=                   x, M_valA=                   x, M_valE=                 256, e_valE=                 256, d_srcA=15, d_srcB=15, e
_dstE= 3, M_dstM=15, M_dstE= 3, W_dstM=15, W_dstE=15

execute_64: E_icode= 3, E_ifun= 0, E_valA=                   x, E_valB=                   x, E_valC=                 512, E_cndnflagin=x, E_stat= 1, E_dstE= 2, E_dstM=15, e_cndnflagout=x, e_valE=
               256, M_valE=                 256, M_cnd=0, M_stat= 1, M_icode= 3, e_dstE= 3, M_dstE= 3, M_dstM=15, M_valA=                   x

memory_64: M_stat= 1, M_icode= 3, M_Cnd=z, M_valE=                 256, M_valA=                   x, M_dstE= 3, M_dstM=15, W_stat= x, W_icode= x, W_valE=                   0, W_valM=
    x, W_dstE=15, W_dstM=15, m_valM=                   x, m_stat= x

writeback_64: W_stat= x, W_icode= x, W_valE=                   0, W_valM=                   x, W_dstE=15, W_dstM=15, R=

             X, Regout=
                                                                                                                           X, M_cnd=0
 ------------------------------------------------------------------------------------------------------------------------------------------------------------------------R0:    9, R1:    8, R2:    7
, R3:    6, R4:   19, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
```

```
                120ns

fetch_64: W_valM=                   x, M_icode= 3, W_icode= 3, M_cnd=0, M_valA=                   x, Jxx_Pred=                   x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, PC=                  22, D_icode= x, D_ifun= x, D_rA= 2, D_rB= 3, D_valC=                 512, D_valP=
                 22, D_stat= 8, f_PC=                   22,

decode_64: E_bubble=0, D_stat= 8, D_icode= x, D_ifun= x, D_rA= 2, D_rB= 3, D_valC=                 512, D_valP=                  22, R=

                             X, E_stat= 1, E_icode= 6, E_ifun= 0, E_valC=                 512, E_valA=                 512, E_valB=                 256, E_dstE= 3, E_dstM=15, E_srcA= 2, E_srcB= 3,
W_valE=                  256, W_valM=                   x, m_valM=                   x, M_valA=                   x, M_valE=                 512, e_valE=                 512, d_srcA= 2, d_srcB= 3, e
_dstE= 2, M_dstM=15, M_dstE= 2, W_dstM=15, W_dstE= 3

execute_64: E_icode= 6, E_ifun= 0, E_valA=                 512, E_valB=                 256, E_valC=                 512, E_cndnflagin=0, E_stat= 1, E_dstE= 3, E_dstM=15, e_cndnflagout=x, e_valE=
               512, M_valE=                 512, M_cnd=0, M_stat= 1, M_icode= 3, e_dstE= 2, M_dstE= 2, M_dstM=15, M_valA=                   x

memory_64: M_stat= 1, M_icode= 3, M_Cnd=z, M_valE=                 512, M_valA=                   x, M_dstE= 2, M_dstM=15, W_stat= 1, W_icode= 3, W_valE=                 256, W_valM=
    x, W_dstE= 3, W_dstM=15, m_valM=                   x, m_stat= 1

writeback_64: W_stat= 1, W_icode= 3, W_valE=                 256, W_valM=                   x, W_dstE= 3, W_dstM=15, R=

             X, Regout=
                                                                                                                           X, M_cnd=0
 ------------------------------------------------------------------------------------------------------------------------------------------------------------------------R0:    9, R1:    8, R2:    7
, R3:    6, R4:   19, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
```

```
                150ns

fetch_64: W_valM=                   x, M_icode= 6, W_icode= 3, M_cnd=0, M_valA=                 512, Jxx_Pred=                   x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, PC=                  22, D_icode= x, D_ifun= x, D_rA= 2, D_rB= 3, D_valC=                 512, D_valP=
                 22, D_stat= 8, f_PC=                   22,

decode_64: E_bubble=0, D_stat= 8, D_icode= x, D_ifun= x, D_rA= 2, D_rB= 3, D_valC=                 512, D_valP=                  22, R=

                             X, E_stat= 8, E_icode= x, E_ifun= x, E_valC=                 512, E_valA=                   7, E_valB=                   6, E_dstE=15, E_dstM=15, E_srcA=15, E_srcB=15,
W_valE=                  512, W_valM=                   x, m_valM=                   x, M_valA=                 512, M_valE=                 768, e_valE=                 768, d_srcA=15, d_srcB=15, e
_dstE= 3, M_dstM=15, M_dstE= 3, W_dstM=15, W_dstE= 2

execute_64: E_icode= x, E_ifun= x, E_valA=                   7, E_valB=                   6, E_valC=                 512, E_cndnflagin=0, E_stat= 8, E_dstE=15, E_dstM=15, e_cndnflagout=0, e_valE=
               768, M_valE=                 768, M_cnd=0, M_stat= 1, M_icode= 6, e_dstE= 3, M_dstE= 3, M_dstM=15, M_valA=                 512

memory_64: M_stat= 1, M_icode= 6, M_Cnd=z, M_valE=                 768, M_valA=                 512, M_dstE= 3, M_dstM=15, W_stat= 1, W_icode= 3, W_valE=                 512, W_valM=
    x, W_dstE= 2, W_dstM=15, m_valM=                   x, m_stat= 1

writeback_64: W_stat= 1, W_icode= 3, W_valE=                 512, W_valM=                   x, W_dstE= 2, W_dstM=15, R=

             X, Regout=
                                                                                                                           X, M_cnd=0
 ------------------------------------------------------------------------------------------------------------------------------------------------------------------------R0:    9, R1:    8, R2:    7
, R3:  256, R4:   19, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
```

```
              180ns

fetch_64: W_valM=                   x, M_icode= x, W_icode= 6, M_cnd=0, M_valA=                   7, Jxx_Pred=                   x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, PC=                  22, D_icode= x, D_ifun= x, D_rA= 2, D_rB= 3, D_valC=                 512, D_valP=
                 22, D_stat= 8, f_PC=                   22,

decode_64: E_bubble=0, D_stat= 8, D_icode= x, D_ifun= x, D_rA= 2, D_rB= 3, D_valC=                 512, D_valP=                  22, R=

                             X, E_stat= 8, E_icode= x, E_ifun= x, E_valC=                 512, E_valA=                   7, E_valB=                   6, E_dstE=15, E_dstM=15, E_srcA=15, E_srcB=15,
W_valE=                 768, W_valM=                   x, m_valM=                   x, M_valA=                   7, M_valE=                   0, e_valE=                   0, d_srcA=15, d_srcB=15, e
_dstE=15, M_dstM=15, M_dstE=15, W_dstM=15, W_dstE= 3

execute_64: E_icode= x, E_ifun= x, E_valA=                   7, E_valB=                   6, E_valC=                 512, E_cndnflagin=0, E_stat= 8, E_dstE=15, E_dstM=15, e_cndnflagout=0, e_valE=
                 0, M_valE=                   0, M_cnd=0, M_stat= 8, M_icode= x, e_dstE=15, M_dstE=15, M_dstM=15, M_valA=                   7

memory_64: M_stat= 8, M_icode= x, M_Cnd=z, M_valE=                   0, M_valA=                   7, M_dstE=15, M_dstM=15, W_stat= 1, W_icode= 6, W_valE=                 768, W_valM=
    x, W_dstE= 3, W_dstM=15, m_valM=                   x, m_stat= 1

writeback_64: W_stat= 1, W_icode= 6, W_valE=                 768, W_valM=                   x, W_dstE= 3, W_dstM=15, R=

             X, Regout=
                                                                                                                                                        X, M_cnd=0
 ---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------R0:     9, R1:     8, R2:  512
, R3:  256, R4:    19, R5:     4, R6:     3, R7:     2, R8:     1, R9:    10, R10:    12, R11:    13, R12:    14, R13:    15, R14:    16
```

```
              210ns

fetch_64: W_valM=                   x, M_icode= x, W_icode= x, M_cnd=0, M_valA=                   7, Jxx_Pred=                   x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, PC=                  22, D_icode= x, D_ifun= x, D_rA= 2, D_rB= 3, D_valC=                 512, D_valP=
                 22, D_stat= 8, f_PC=                   22,

decode_64: E_bubble=0, D_stat= 8, D_icode= x, D_ifun= x, D_rA= 2, D_rB= 3, D_valC=                 512, D_valP=                  22, R=

                             X, E_stat= 8, E_icode= x, E_ifun= x, E_valC=                 512, E_valA=                   7, E_valB=                   6, E_dstE=15, E_dstM=15, E_srcA=15, E_srcB=15,
W_valE=                   0, W_valM=                   x, m_valM=                   x, M_valA=                   7, M_valE=                   0, e_valE=                   0, d_srcA=15, d_srcB=15, e
_dstE=15, M_dstM=15, M_dstE=15, W_dstM=15, W_dstE=15

execute_64: E_icode= x, E_ifun= x, E_valA=                   7, E_valB=                   6, E_valC=                 512, E_cndnflagin=0, E_stat= 8, E_dstE=15, E_dstM=15, e_cndnflagout=0, e_valE=
                 0, M_valE=                   0, M_cnd=0, M_stat= 8, M_icode= x, e_dstE=15, M_dstE=15, M_dstM=15, M_valA=                   7

memory_64: M_stat= 8, M_icode= x, M_Cnd=z, M_valE=                   0, M_valA=                   7, M_dstE=15, M_dstM=15, W_stat= 8, W_icode= x, W_valE=                   0, W_valM=
    x, W_dstE=15, W_dstM=15, m_valM=                   x, m_stat= 8

writeback_64: W_stat= 8, W_icode= x, W_valE=                   0, W_valM=                   x, W_dstE=15, W_dstM=15, R=

             X, Regout=
                                                                                                                                                        X, M_cnd=0
 ---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------R0:     9, R1:     8, R2:  512
, R3:  768, R4:    19, R5:     4, R6:     3, R7:     2, R8:     1, R9:    10, R10:    12, R11:    13, R12:    14, R13:    15, R14:    16
```

2)

```
irmovq $0x100, %rbx
rmmovq %rbx, 3(%rax)
mrmovq %rcx, 3(%rax)
```

```
              30ns

PC:                    10
fetch_64: W_valM=                   x, M_icode= x, W_icode= x, M_cnd=0, M_valA=                   x, Jxx_Pred=                   x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=01000000001100000000001100000000000000000000000000000000000000000000000000000000, PC=                  10, D_icode= 3, D_ifun= 0, D_rA=15, D_rB= 3, D_valC=                 256, D_valP=
                 10, D_stat= 1, f_PC=                   10,

decode_64: E_bubble=0, D_stat= 1, D_icode= 3, D_ifun= 0, D_rA=15, D_rB= 3, D_valC=                 256, D_valP=                  10, R=

                             X, E_stat= x, E_icode= x, E_ifun= x, E_valC=                   x, E_valA=                   x, E_valB=                   x, E_dstE=15, E_dstM=15, E_srcA=15, E_srcB=15, W_v
alE=                   x, W_valM=                   x, m_valM=                   x, M_valA=                   x, M_valE=                   0, e_valE=                   0, d_srcA=15, d_srcB=15, e_dst
E= x, M_dstM= x, M_dstE= x, W_dstM= x, W_dstE= x

execute_64: E_icode= x, E_ifun= x, E_valA=                   x, E_valB=                   x, E_valC=                   x, E_cndnflagin=x, E_stat= x, E_dstE=15, E_dstM=15, e_cndnflagout=x, e_valE=
                 0, M_valE=                   0, M_cnd=0, M_stat= x, M_icode= x, e_dstE= x, M_dstE= x, M_dstM= x, M_valA=                   x

memory_64: M_stat= x, M_icode= x, M_Cnd=z, M_valE=                   0, M_valA=                   x, M_dstE= x, M_dstM= x, W_stat= x, W_icode= x, W_valE=                   x, W_valM=
   x, W_dstE= x, W_dstM= x, m_valM=                   x, m_stat= x

writeback_64: W_stat= x, W_icode= x, W_valE=                   x, W_valM=                   x, W_dstE= x, W_dstM= x, R=

           X, Regout=
                                                                                                                                                        X, M_cnd=0
 ---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
R0:     9, R1:     8, R2:     7, R3:     6, R4:    19, R5:     4, R6:     3, R7:     2, R8:     1, R9:    10, R10:    12, R11:    13, R12:    14, R13:    15, R14:    16
```

```
                60ns

PC:                20
fetch_64: W_valM=                  x, M_icode= x, W_icode= x, M_cnd=0, M_valA=                  x, Jxx_Pred=                  x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=0101000000010000000001100000000000000000000000000000000000000000000000000000000000, PC=                 20, D_icode= 4, D_ifun= 0, D_rA= 3, D_rB= 0, D_valC=                  3, D_valP=
             20, D_stat= 1, f_PC=                 20,


decode_64: E_bubble=0, D_stat= 1, D_icode= 4, D_ifun= 0, D_rA= 3, D_rB= 0, D_valC=                  3, D_valP=                 20, R=

                     X, E_stat= 1, E_icode= 3, E_ifun= 0, E_valC=                256, E_valA=                  x, E_valB=                  x, E_dstE= 3, E_dstM=15, E_srcA=15, E_srcB=15, W_v
alE=                 0, W_valM=                  x, m_valM=                  x, M_valA=                  x, M_valE=                  0, e_valE=                  0, d_srcA=15, d_srcB=15, e_dst
E=15, M_dstM=15, M_dstE=15, W_dstM= x, W_dstE= x


execute_64: E_icode= 3, E_ifun= 0, E_valA=                  x, E_valB=                  x, E_valC=                256, E_cndnflagin=x, E_stat= 1, E_dstE= 3, E_dstM=15, e_cndnflagout=x, e_valE=
             0, M_valE=                  0, M_cnd=0, M_stat= x, M_icode= x, e_dstE=15, M_dstE=15, M_dstM=15, M_valA=                  x


memory_64: M_stat= x, M_icode= x, M_Cnd=z, M_valE=                  0, M_valA=                  x, M_dstE=15, M_dstM=15, W_stat= x, W_icode= x, W_valE=                  0, W_valM=
   x, W_dstE= x, W_dstM= x, m_valM=                  x, m_stat= x


writeback_64: W_stat= x, W_icode= x, W_valE=                  0, W_valM=                  x, W_dstE= x, W_dstM= x, R=

          X, Regout=
                                                                                                                                X, M_cnd=0
 ------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
R0:    9, R1:    8, R2:    7, R3:    6, R4:   19, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
```

```
                90ns

PC:                30
fetch_64: W_valM=                  x, M_icode= 3, W_icode= x, M_cnd=0, M_valA=                  x, Jxx_Pred=                  x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, PC=                 30, D_icode= 5, D_ifun= 0, D_rA= 1, D_rB= 0, D_valC=                  3, D_valP=
             30, D_stat= 1, f_PC=                 30,


decode_64: E_bubble=0, D_stat= 1, D_icode= 5, D_ifun= 0, D_rA= 1, D_rB= 0, D_valC=                  3, D_valP=                 30, R=

                     X, E_stat= 1, E_icode= 4, E_ifun= 0, E_valC=                  3, E_valA=                256, E_valB=                  9, E_dstE=15, E_dstM=15, E_srcA= 3, E_srcB= 0, W_v
alE=                 0, W_valM=                  x, m_valM=                  x, M_valA=                  x, M_valE=                256, e_valE=                256, d_srcA= 3, d_srcB= 0, e_dst
E= 3, M_dstM=15, M_dstE= 3, W_dstM=15, W_dstE=15


execute_64: E_icode= 4, E_ifun= 0, E_valA=                256, E_valB=                  9, E_valC=                  3, E_cndnflagin=x, E_stat= 1, E_dstE=15, E_dstM=15, e_cndnflagout=x, e_valE=
            256, M_valE=                256, M_cnd=0, M_stat= 1, M_icode= 3, e_dstE= 3, M_dstE= 3, M_dstM=15, M_valA=                  x


memory_64: M_stat= 1, M_icode= 3, M_Cnd=z, M_valE=                256, M_valA=                  x, M_dstE= 3, M_dstM=15, W_stat= x, W_icode= x, W_valE=                  0, W_valM=
   x, W_dstE=15, W_dstM=15, m_valM=                  x, m_stat= x


writeback_64: W_stat= x, W_icode= x, W_valE=                  0, W_valM=                  x, W_dstE=15, W_dstM=15, R=

          X, Regout=
                                                                                                                                X, M_cnd=0
 ------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
R0:    9, R1:    8, R2:    7, R3:    6, R4:   19, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
```

```
               120ns

PC:                30
fetch_64: W_valM=                  x, M_icode= 4, W_icode= 3, M_cnd=0, M_valA=                256, Jxx_Pred=                  x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, PC=                 30, D_icode= x, D_ifun= x, D_rA= 1, D_rB= 0, D_valC=                  3, D_valP=
             30, D_stat= 8, f_PC=                 30,


decode_64: E_bubble=0, D_stat= 8, D_icode= x, D_ifun= x, D_rA= 1, D_rB= 0, D_valC=                  3, D_valP=                 30, R=

                     X, E_stat= 1, E_icode= 5, E_ifun= 0, E_valC=                  3, E_valA=                  6, E_valB=                  9, E_dstE=15, E_dstM= 1, E_srcA=15, E_srcB= 0, W_v
alE=               256, W_valM=                  x, m_valM=                  x, M_valA=                256, M_valE=                 12, e_valE=                 12, d_srcA=15, d_srcB= 0, e_dst
E=15, M_dstM=15, M_dstE=15, W_dstM=15, W_dstE= 3


execute_64: E_icode= 5, E_ifun= 0, E_valA=                  6, E_valB=                  9, E_valC=                  3, E_cndnflagin=0, E_stat= 1, E_dstE=15, E_dstM= 1, e_cndnflagout=x, e_valE=
            12, M_valE=                 12, M_cnd=0, M_stat= 1, M_icode= 4, e_dstE=15, M_dstE=15, M_dstM=15, M_valA=                256


memory_64: M_stat= 1, M_icode= 4, M_Cnd=z, M_valE=                 12, M_valA=                256, M_dstE=15, M_dstM=15, W_stat= 1, W_icode= 3, W_valE=                256, W_valM=
   x, W_dstE= 3, W_dstM=15, m_valM=                  x, m_stat= 1


writeback_64: W_stat= 1, W_icode= 3, W_valE=                256, W_valM=                  x, W_dstE= 3, W_dstM=15, R=

          X, Regout=
                                                                                                                                X, M_cnd=0
 ------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
R0:    9, R1:    8, R2:    7, R3:    6, R4:   19, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
```

```
               150ns

PC:                30
fetch_64: W_valM=                  x, M_icode= 5, W_icode= 4, M_cnd=0, M_valA=                  6, Jxx_Pred=                  x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, PC=                 30, D_icode= x, D_ifun= x, D_rA= 1, D_rB= 0, D_valC=                  3, D_valP=
             30, D_stat= 8, f_PC=                 30,


decode_64: E_bubble=0, D_stat= 8, D_icode= x, D_ifun= x, D_rA= 1, D_rB= 0, D_valC=                  3, D_valP=                 30, R=

                     X, E_stat= 8, E_icode= x, E_ifun= x, E_valC=                  3, E_valA=                  6, E_valB=                  9, E_dstE=15, E_dstM=15, E_srcA=15, E_srcB=15, W_v
alE=                12, W_valM=                  x, m_valM=                  x, M_valA=                  6, M_valE=                 12, e_valE=                 12, d_srcA=15, d_srcB=15, e_dst
E=15, M_dstM= 1, M_dstE=15, W_dstM=15, W_dstE=15


execute_64: E_icode= x, E_ifun= x, E_valA=                  6, E_valB=                  9, E_valC=                  3, E_cndnflagin=0, E_stat= 8, E_dstE=15, E_dstM=15, e_cndnflagout=x, e_valE=
            12, M_valE=                 12, M_cnd=0, M_stat= 1, M_icode= 5, e_dstE=15, M_dstE=15, M_dstM= 1, M_valA=                  6


memory_64: M_stat= 1, M_icode= 5, M_Cnd=z, M_valE=                 12, M_valA=                  6, M_dstE=15, M_dstM= 1, W_stat= 1, W_icode= 4, W_valE=                 12, W_valM=
   x, W_dstE=15, W_dstM=15, m_valM=                  x, m_stat= 1


writeback_64: W_stat= 1, W_icode= 4, W_valE=                 12, W_valM=                  x, W_dstE=15, W_dstM=15, R=

          X, Regout=
                                                                                                                                X, M_cnd=0
 ------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
R0:    9, R1:    8, R2:    7, R3:  256, R4:   19, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
```

```
                180ns


PC:                 30
fetch_64: W_valM=             256, M_icode= x, W_icode= 5, M_cnd=0, M_valA=                6, Jxx_Pred=                x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, PC=              30, D_icode= x, D_ifun= x, D_rA= 1, D_rB= 0, D_valC=                3, D_valP=
              30, D_stat= 8, f_PC=                30,


decode_64: E_bubble=0, D_stat= 8, D_icode= x, D_ifun= x, D_rA= 1, D_rB= 0, D_valC=                3, D_valP=                30, R=

                     X, E_stat= 8, E_icode= x, E_ifun= x, E_valC=                3, E_valA=                6, E_valB=                9, E_dstE=15, E_dstM=15, E_srcA=15, E_srcB=15, W_v
alE=              12, W_valM=              256, m_valM=              256, M_valA=              6, M_valE=              0, e_valE=                0, d_srcA=15, d_srcB=15, e_dst
E=15, M_dstM=15, M_dstE=15, W_dstM= 1, W_dstE=15


execute_64: E_icode= x, E_ifun= x, E_valA=              6, E_valB=              9, E_valC=              3, E_cndnflagin=0, E_stat= 8, E_dstE=15, E_dstM=15, e_cndnflagout=x, e_valE=
              0, M_valE=              0, M_cnd=0, M_stat= 8, M_icode= x, e_dstE=15, M_dstE=15, M_dstM=15, M_valA=              6


memory_64: M_stat= 8, M_icode= x, M_Cnd=z, M_valE=                0, M_valA=                6, M_dstE=15, M_dstM=15, W_stat= 1, W_icode= 5, W_valE=               12, W_valM=
 256, W_dstE=15, W_dstM= 1, m_valM=              256, m_stat= 1


writeback_64: W_stat= 1, W_icode= 5, W_valE=               12, W_valM=              256, W_dstE=15, W_dstM= 1, R=

          X, Regout=
                                                                                                                  X, M_cnd=0
 ------------------------------------------------------------------------------------------------------------------------------------------------------
R0:    9, R1:    8, R2:    7, R3:  256, R4:   19, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
```

```
                195ns


PC:                 30
fetch_64: W_valM=             256, M_icode= x, W_icode= 5, M_cnd=0, M_valA=                6, Jxx_Pred=                x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, PC=              30, D_icode= x, D_ifun= x, D_rA= 1, D_rB= 0, D_valC=                3, D_valP=
              30, D_stat= 8, f_PC=                30,


decode_64: E_bubble=0, D_stat= 8, D_icode= x, D_ifun= x, D_rA= 1, D_rB= 0, D_valC=                3, D_valP=                30, R=

                     X, E_stat= 8, E_icode= x, E_ifun= x, E_valC=                3, E_valA=                6, E_valB=                9, E_dstE=15, E_dstM=15, E_srcA=15, E_srcB=15, W_v
alE=              12, W_valM=              256, m_valM=              256, M_valA=              6, M_valE=              0, e_valE=                0, d_srcA=15, d_srcB=15, e_dst
E=15, M_dstM=15, M_dstE=15, W_dstM= 1, W_dstE=15


execute_64: E_icode= x, E_ifun= x, E_valA=              6, E_valB=              9, E_valC=              3, E_cndnflagin=0, E_stat= 8, E_dstE=15, E_dstM=15, e_cndnflagout=x, e_valE=
              0, M_valE=              0, M_cnd=0, M_stat= 8, M_icode= x, e_dstE=15, M_dstE=15, M_dstM=15, M_valA=              6


memory_64: M_stat= 8, M_icode= x, M_Cnd=z, M_valE=                0, M_valA=                6, M_dstE=15, M_dstM=15, W_stat= 1, W_icode= 5, W_valE=               12, W_valM=
 256, W_dstE=15, W_dstM= 1, m_valM=              256, m_stat= 8


writeback_64: W_stat= 1, W_icode= 5, W_valE=               12, W_valM=              256, W_dstE=15, W_dstM= 1, R=

          X, Regout=
                                                                                                                  X, M_cnd=0
 ------------------------------------------------------------------------------------------------------------------------------------------------------
R0:    9, R1:  256, R2:    7, R3:  256, R4:   19, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
```

3)

```
1   irmovq $4, %rax
2   irmovq $5, %rbx
3   subq %rax, %rbx
4   jg L2
5   irmovq $1, %r8
6   jmp end
7   L2: irmovq $2, %r8
8   end:
```

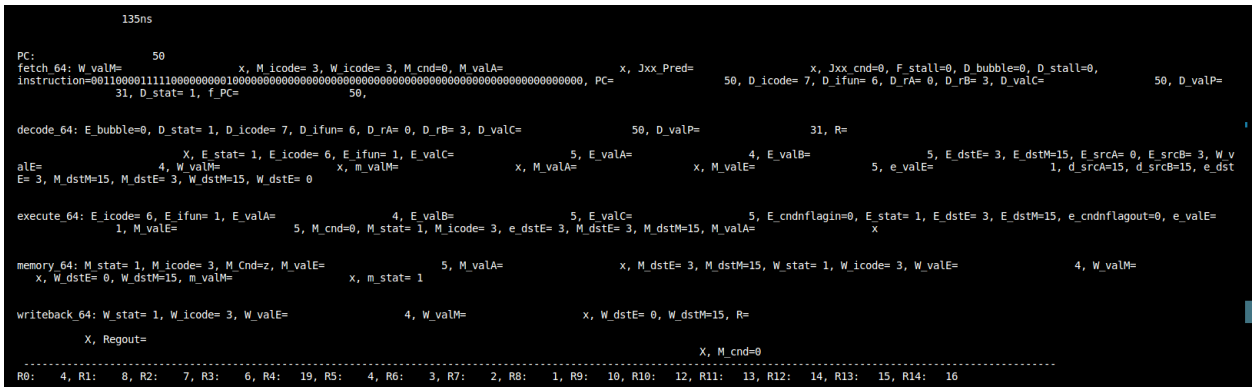
```
                15ns


PC:                 0
fetch_64: W_valM=             x, M_icode= x, W_icode= x, M_cnd=0, M_valA=                x, Jxx_Pred=                x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=00110000111100000000100000000000000000000000000000000000000000000000000000, PC=              0, D_icode= x, D_ifun= x, D_rA= x, D_rB= x, D_valC=                x, D_valP=
              x, D_stat= x, f_PC=                x,


decode_64: E_bubble=0, D_stat= x, D_icode= x, D_ifun= x, D_rA= x, D_rB= x, D_valC=                x, D_valP=                x, R=

                     X, E_stat= x, E_icode= x, E_ifun= x, E_valC=                x, E_valA=                x, E_valB=                x, E_dstE= x, E_dstM= x, E_srcA= x, E_srcB= x, W_v
alE=              x, W_valM=              x, m_valM=              x, M_valA=              x, M_valE=              x, e_valE=                0, d_srcA=15, d_srcB=15, e_dst
E= x, M_dstM= x, M_dstE= x, W_dstM= x, W_dstE= x


execute_64: E_icode= x, E_ifun= x, E_valA=                x, E_valB=                x, E_valC=                x, E_cndnflagin=x, E_stat= x, E_dstE= x, E_dstM= x, e_cndnflagout=x, e_valE=
              0, M_valE=              x, M_cnd=0, M_stat= x, M_icode= x, e_dstE= x, M_dstE= x, M_dstM= x, M_valA=              x


memory_64: M_stat= x, M_icode= x, M_Cnd=z, M_valE=                x, M_valA=                x, M_dstE= x, M_dstM= x, W_stat= x, W_icode= x, W_valE=                x, W_valM=
   x, W_dstE= x, W_dstM= x, m_valM=              x, m_stat= x


writeback_64: W_stat= x, W_icode= x, W_valE=               x, W_valM=              x, W_dstE= x, W_dstM= x, R=

          X, Regout=
                                                                                                                  X, M_cnd=0
 ------------------------------------------------------------------------------------------------------------------------------------------------------
R0:    9, R1:    8, R2:    7, R3:    6, R4:   19, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
```

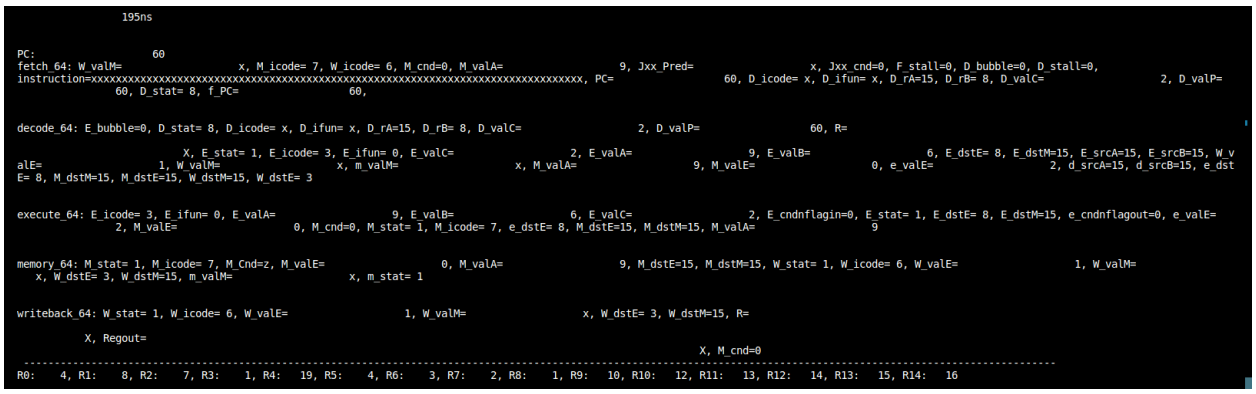
```
                 45ns


PC:                 10
fetch_64: W_valM=                    x, M_icode= x, W_icode= x, M_cnd=0, M_valA=                    x, Jxx_Pred=                    x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=00110000111100110000010100000000000000000000000000000000000000000000000000000000, PC=                    10, D_icode= 3, D_ifun= 0, D_rA=15, D_rB= 0, D_valC=                    4, D_valP=
               10, D_stat= 1, f_PC=                    10,


decode_64: E_bubble=0, D_stat= 1, D_icode= 3, D_ifun= 0, D_rA=15, D_rB= 0, D_valC=                    4, D_valP=                    10, R=

                          X, E_stat= x, E_icode= x, E_ifun= x, E_valC=                    x, E_valA=                    x, E_valB=                    x, E_dstE=15, E_dstM=15, E_srcA=15, E_srcB=15, W_v
alE=                    x, W_valM=                    x, m_valM=                    x, M_valA=                    x, M_valE=                    0, e_valE=                    0, d_srcA=15, d_srcB=15, e_dst
E=15, M_dstM= x, M_dstE= x, W_dstM= x, W_dstE= x


execute_64: E_icode= x, E_ifun= x, E_valA=                    x, E_valB=                    x, E_valC=                    x, E_cndnflagin=x, E_stat= x, E_dstE=15, E_dstM=15, e_cndnflagout=x, e_valE=
               0, M_valE=                    0, M_cnd=0, M_stat= x, M_icode= x, e_dstE=15, M_dstE= x, M_dstM= x, M_valA=                    x


memory_64: M_stat= x, M_icode= x, M_Cnd=z, M_valE=                    0, M_valA=                    x, M_dstE= x, M_dstM= x, W_stat= x, W_icode= x, W_valE=                    x, W_valM=
   x, W_dstE= x, W_dstM= x, m_valM=                    x, m_stat= x


writeback_64: W_stat= x, W_icode= x, W_valE=                    x, W_valM=                    x, W_dstE= x, W_dstM= x, R=

          X, Regout=
                                                                                                          X, M_cnd=0
 --------------------------------------------------------------------------------------------------------------------------------------------------------------
R0:    9, R1:    8, R2:    7, R3:    6, R4:   19, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
```

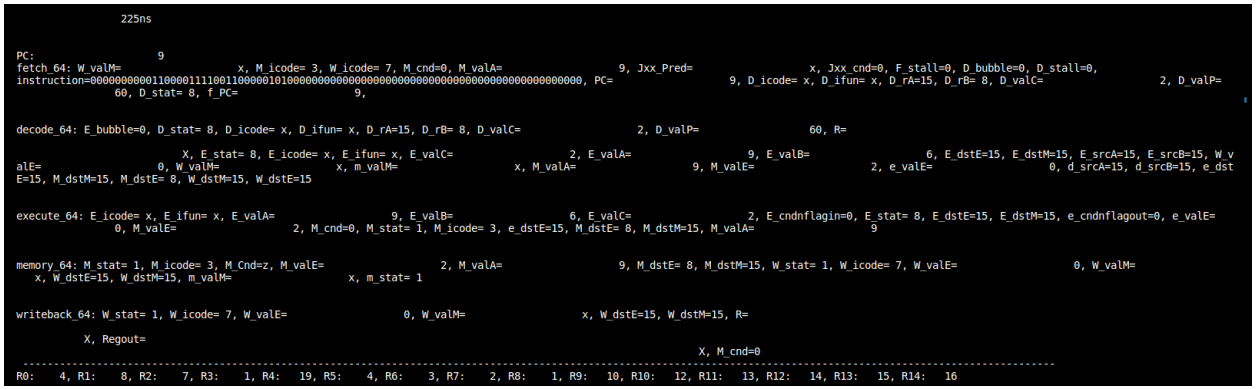
```
                 75ns


PC:                 20
fetch_64: W_valM=                    x, M_icode= x, W_icode= x, M_cnd=0, M_valA=                    x, Jxx_Pred=                    x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=01100001000000110111011000110010000000000000000000000000000000000000000000000000, PC=                    20, D_icode= 3, D_ifun= 0, D_rA=15, D_rB= 3, D_valC=                    5, D_valP=
               20, D_stat= 1, f_PC=                    20,


decode_64: E_bubble=0, D_stat= 1, D_icode= 3, D_ifun= 0, D_rA=15, D_rB= 3, D_valC=                    5, D_valP=                    20, R=

                          X, E_stat= 1, E_icode= 3, E_ifun= 0, E_valC=                    4, E_valA=                    x, E_valB=                    x, E_dstE= 0, E_dstM=15, E_srcA=15, E_srcB=15, W_v
alE=                    0, W_valM=                    x, m_valM=                    x, M_valA=                    x, M_valE=                    0, e_valE=                    4, d_srcA=15, d_srcB=15, e_dst
E= 0, M_dstM=15, M_dstE=15, W_dstM= x, W_dstE= x


execute_64: E_icode= 3, E_ifun= 0, E_valA=                    x, E_valB=                    x, E_valC=                    4, E_cndnflagin=x, E_stat= 1, E_dstE= 0, E_dstM=15, e_cndnflagout=x, e_valE=
               4, M_valE=                    0, M_cnd=0, M_stat= x, M_icode= x, e_dstE= 0, M_dstE=15, M_dstM=15, M_valA=                    x


memory_64: M_stat= x, M_icode= x, M_Cnd=z, M_valE=                    0, M_valA=                    x, M_dstE=15, M_dstM=15, W_stat= x, W_icode= x, W_valE=                    0, W_valM=
   x, W_dstE= x, W_dstM= x, m_valM=                    x, m_stat= x


writeback_64: W_stat= x, W_icode= x, W_valE=                    0, W_valM=                    x, W_dstE= x, W_dstM= x, R=

          X, Regout=
                                                                                                          X, M_cnd=0
 --------------------------------------------------------------------------------------------------------------------------------------------------------------
R0:    9, R1:    8, R2:    7, R3:    6, R4:   19, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
```

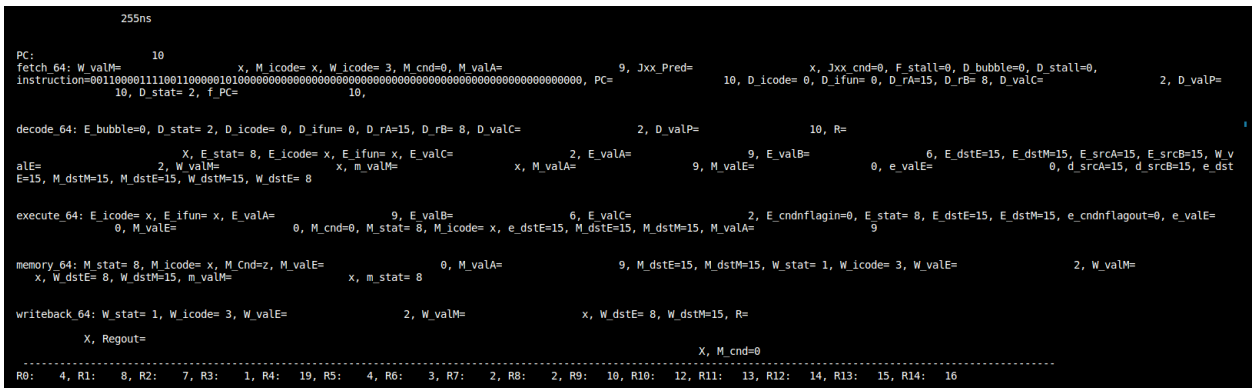
```
                105ns


PC:                 22
fetch_64: W_valM=                    x, M_icode= 3, W_icode= x, M_cnd=0, M_valA=                    x, Jxx_Pred=                    x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=01110110001100100000000000000000000000000000000000000000000000000000000000110000, PC=                    22, D_icode= 6, D_ifun= 1, D_rA= 0, D_rB= 3, D_valC=                    5, D_valP=
               22, D_stat= 1, f_PC=                    22,


decode_64: E_bubble=0, D_stat= 1, D_icode= 6, D_ifun= 1, D_rA= 0, D_rB= 3, D_valC=                    5, D_valP=                    22, R=

                          X, E_stat= 1, E_icode= 3, E_ifun= 0, E_valC=                    5, E_valA=                    x, E_valB=                    x, E_dstE= 3, E_dstM=15, E_srcA=15, E_srcB=15, W_v
alE=                    0, W_valM=                    x, m_valM=                    x, M_valA=                    x, M_valE=                    4, e_valE=                    5, d_srcA= 0, d_srcB= 3, e_dst
E= 3, M_dstM=15, M_dstE= 0, W_dstM=15, W_dstE=15


execute_64: E_icode= 3, E_ifun= 0, E_valA=                    x, E_valB=                    x, E_valC=                    5, E_cndnflagin=0, E_stat= 1, E_dstE= 3, E_dstM=15, e_cndnflagout=x, e_valE=
               5, M_valE=                    4, M_cnd=0, M_stat= 1, M_icode= 3, e_dstE= 3, M_dstE= 0, M_dstM=15, M_valA=                    x


memory_64: M_stat= 1, M_icode= 3, M_Cnd=z, M_valE=                    4, M_valA=                    x, M_dstE= 0, M_dstM=15, W_stat= x, W_icode= x, W_valE=                    0, W_valM=
   x, W_dstE=15, W_dstM=15, m_valM=                    x, m_stat= 1


writeback_64: W_stat= x, W_icode= x, W_valE=                    0, W_valM=                    x, W_dstE=15, W_dstM=15, R=

          X, Regout=
                                                                                                          X, M_cnd=0
 --------------------------------------------------------------------------------------------------------------------------------------------------------------
R0:    9, R1:    8, R2:    7, R3:    6, R4:   19, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
```

```
                165ns


PC:                 60
fetch_64: W_valM=                    x, M_icode= 6, W_icode= 3, M_cnd=1, M_valA=                    4, Jxx_Pred=                    x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, PC=                    60, D_icode= 3, D_ifun= 0, D_rA=15, D_rB= 8, D_valC=                    2, D_valP=
               60, D_stat= 1, f_PC=                    60,


decode_64: E_bubble=0, D_stat= 1, D_icode= 3, D_ifun= 0, D_rA=15, D_rB= 8, D_valC=                    2, D_valP=                    60, R=

                          X, E_stat= 1, E_icode= 7, E_ifun= 6, E_valC=                   50, E_valA=                    9, E_valB=                    6, E_dstE=15, E_dstM=15, E_srcA=15, E_srcB=15, W_v
alE=                    5, W_valM=                    x, m_valM=                    x, M_valA=                    4, M_valE=                    1, e_valE=                    0, d_srcA=15, d_srcB=15, e_dst
E=15, M_dstM=15, M_dstE= 3, W_dstM=15, W_dstE= 3


execute_64: E_icode= 7, E_ifun= 6, E_valA=                    9, E_valB=                    6, E_valC=                   50, E_cndnflagin=0, E_stat= 1, E_dstE=15, E_dstM=15, e_cndnflagout=0, e_valE=
               0, M_valE=                    1, M_cnd=1, M_stat= 1, M_icode= 6, e_dstE=15, M_dstE= 3, M_dstM=15, M_valA=                    4


memory_64: M_stat= 1, M_icode= 6, M_Cnd=z, M_valE=                    1, M_valA=                    4, M_dstE= 3, M_dstM=15, W_stat= 1, W_icode= 3, W_valE=                    5, W_valM=
   x, W_dstE= 3, W_dstM=15, m_valM=                    x, m_stat= 1


writeback_64: W_stat= 1, W_icode= 3, W_valE=                    5, W_valM=                    x, W_dstE= 3, W_dstM=15, R=

          X, Regout=
                                                                                                          X, M_cnd=1
 --------------------------------------------------------------------------------------------------------------------------------------------------------------
R0:    4, R1:    8, R2:    7, R3:    5, R4:   19, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
```


135ns
PC: 50
fetch_64: W_valM= x, M_icode= 3, W_icode= 3, M_cnd=0, M_valA= x, Jxx_Pred= x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=0011000011111000000001000000000000000000000000000000000000000000000000000000000000000000, PC= 50, D_icode= 7, D_ifun= 6, D_rA= 0, D_rB= 3, D_valC= 50, D_valP=
31, D_stat= 1, f_PC= 50,
decode_64: E_bubble=0, D_stat= 1, D_icode= 7, D_ifun= 6, D_rA= 0, D_rB= 3, D_valC= 50, D_valP= 31, R=
X, E_stat= 1, E_icode= 6, E_ifun= 1, E_valC= 5, E_valA= 4, E_valB= 5, E_dstE= 3, E_dstM=15, E_srcA= 0, E_srcB= 3, W_v
alE= 4, W_valM= x, m_valM= x, M_valA= x, M_valE= 5, e_valE= 1, d_srcA=15, d_srcB=15, e_dst
E= 3, M_dstM=15, M_dstE= 3, W_dstM=15, W_dstE= 0
execute_64: E_icode= 6, E_ifun= 1, E_valA= 4, E_valB= 5, E_valC= 5, E_cndnflagin=0, E_stat= 1, E_dstE= 3, E_dstM=15, e_cndnflagout=0, e_valE=
1, M_valE= 5, M_cnd=0, M_stat= 1, M_icode= 3, e_dstE= 3, M_dstE= 3, M_dstM=15, M_valA= x
memory_64: M_stat= 1, M_icode= 3, M_Cnd=z, M_valE= 5, M_valA= x, M_dstE= 3, M_dstM=15, W_stat= 1, W_icode= 3, W_valE= 4, W_valM=
x, W_dstE= 0, W_dstM=15, m_valM= x, m_stat= 1
writeback_64: W_stat= 1, W_icode= 3, W_valE= 4, W_valM= x, W_dstE= 0, W_dstM=15, R=
X, Regout=
X, M_cnd=0
R0: 4, R1: 8, R2: 7, R3: 6, R4: 19, R5: 4, R6: 3, R7: 2, R8: 1, R9: 10, R10: 12, R11: 13, R12: 14, R13: 15, R14: 16


195ns
PC: 60
fetch_64: W_valM= x, M_icode= 7, W_icode= 6, M_cnd=0, M_valA= 9, Jxx_Pred= x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, PC= 60, D_icode= x, D_ifun= x, D_rA=15, D_rB= 8, D_valC= 2, D_valP=
60, D_stat= 8, f_PC= 60,
decode_64: E_bubble=0, D_stat= 8, D_icode= x, D_ifun= x, D_rA=15, D_rB= 8, D_valC= 2, D_valP= 60, R=
X, E_stat= 1, E_icode= 3, E_ifun= 0, E_valC= 2, E_valA= 9, E_valB= 6, E_dstE= 8, E_dstM=15, E_srcA=15, E_srcB=15, W_v
alE= 1, W_valM= x, m_valM= x, M_valA= 9, M_valE= 0, e_valE= 2, d_srcA=15, d_srcB=15, e_dst
E= 8, M_dstM=15, M_dstE=15, W_dstM=15, W_dstE= 3
execute_64: E_icode= 3, E_ifun= 0, E_valA= 9, E_valB= 6, E_valC= 2, E_cndnflagin=0, E_stat= 1, E_dstE= 8, E_dstM=15, e_cndnflagout=0, e_valE=
2, M_valE= 0, M_cnd=0, M_stat= 1, M_icode= 7, e_dstE= 8, M_dstE=15, M_dstM=15, M_valA= 9
memory_64: M_stat= 1, M_icode= 7, M_Cnd=z, M_valE= 0, M_valA= 9, M_dstE=15, M_dstM=15, W_stat= 1, W_icode= 6, W_valE= 1, W_valM=
x, W_dstE= 3, W_dstM=15, m_valM= x, m_stat= 1
writeback_64: W_stat= 1, W_icode= 6, W_valE= 1, W_valM= x, W_dstE= 3, W_dstM=15, R=
X, Regout=
X, M_cnd=0
R0: 4, R1: 8, R2: 7, R3: 1, R4: 19, R5: 4, R6: 3, R7: 2, R8: 1, R9: 10, R10: 12, R11: 13, R12: 14, R13: 15, R14: 16


225ns
PC: 9
fetch_64: W_valM= x, M_icode= 3, W_icode= 7, M_cnd=0, M_valA= 9, Jxx_Pred= x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=0000000000011000011110011000001010000000000000000000000000000000000000000000000000000000, PC= 9, D_icode= x, D_ifun= x, D_rA=15, D_rB= 8, D_valC= 2, D_valP=
60, D_stat= 8, f_PC= 9,
decode_64: E_bubble=0, D_stat= 8, D_icode= x, D_ifun= x, D_rA=15, D_rB= 8, D_valC= 2, D_valP= 60, R=
X, E_stat= 8, E_icode= x, E_ifun= x, E_valC= 2, E_valA= 9, E_valB= 6, E_dstE=15, E_dstM=15, E_srcA=15, E_srcB=15, W_v
alE= 0, W_valM= x, m_valM= x, M_valA= 9, M_valE= 2, e_valE= 0, d_srcA=15, d_srcB=15, e_dst
E=15, M_dstM=15, M_dstE= 8, W_dstM=15, W_dstE=15
execute_64: E_icode= x, E_ifun= x, E_valA= 9, E_valB= 6, E_valC= 2, E_cndnflagin=0, E_stat= 8, E_dstE=15, E_dstM=15, e_cndnflagout=0, e_valE=
0, M_valE= 2, M_cnd=0, M_stat= 1, M_icode= 3, e_dstE=15, M_dstE= 8, M_dstM=15, M_valA= 9
memory_64: M_stat= 1, M_icode= 3, M_Cnd=z, M_valE= 2, M_valA= 9, M_dstE= 8, M_dstM=15, W_stat= 1, W_icode= 7, W_valE= 0, W_valM=
x, W_dstE=15, W_dstM=15, m_valM= x, m_stat= 1
writeback_64: W_stat= 1, W_icode= 7, W_valE= 0, W_valM= x, W_dstE=15, W_dstM=15, R=
X, Regout=
X, M_cnd=0
R0: 4, R1: 8, R2: 7, R3: 1, R4: 19, R5: 4, R6: 3, R7: 2, R8: 1, R9: 10, R10: 12, R11: 13, R12: 14, R13: 15, R14: 16


255ns
PC: 10
fetch_64: W_valM= x, M_icode= x, W_icode= 3, M_cnd=0, M_valA= 9, Jxx_Pred= x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=0011000011110011000001010000000000000000000000000000000000000000000000000000000000000000, PC= 10, D_icode= 0, D_ifun= 0, D_rA=15, D_rB= 8, D_valC= 2, D_valP=
10, D_stat= 2, f_PC= 10,
decode_64: E_bubble=0, D_stat= 2, D_icode= 0, D_ifun= 0, D_rA=15, D_rB= 8, D_valC= 2, D_valP= 10, R=
X, E_stat= 8, E_icode= x, E_ifun= x, E_valC= 2, E_valA= 9, E_valB= 6, E_dstE=15, E_dstM=15, E_srcA=15, E_srcB=15, W_v
alE= 2, W_valM= x, m_valM= x, M_valA= 9, M_valE= 0, e_valE= 0, d_srcA=15, d_srcB=15, e_dst
E=15, M_dstM=15, M_dstE=15, W_dstM=15, W_dstE= 8
execute_64: E_icode= x, E_ifun= x, E_valA= 9, E_valB= 6, E_valC= 2, E_cndnflagin=0, E_stat= 8, E_dstE=15, E_dstM=15, e_cndnflagout=0, e_valE=
0, M_valE= 0, M_cnd=0, M_stat= 8, M_icode= x, e_dstE=15, M_dstE=15, M_dstM=15, M_valA= 9
memory_64: M_stat= 8, M_icode= x, M_Cnd=z, M_valE= 0, M_valA= 9, M_dstE=15, M_dstM=15, W_stat= 1, W_icode= 3, W_valE= 2, W_valM=
x, W_dstE= 8, W_dstM=15, m_valM= x, m_stat= 8
writeback_64: W_stat= 1, W_icode= 3, W_valE= 2, W_valM= x, W_dstE= 8, W_dstM=15, R=
X, Regout=
X, M_cnd=0
R0: 4, R1: 8, R2: 7, R3: 1, R4: 19, R5: 4, R6: 3, R7: 2, R8: 2, R9: 10, R10: 12, R11: 13, R12: 14, R13: 15, R14: 16

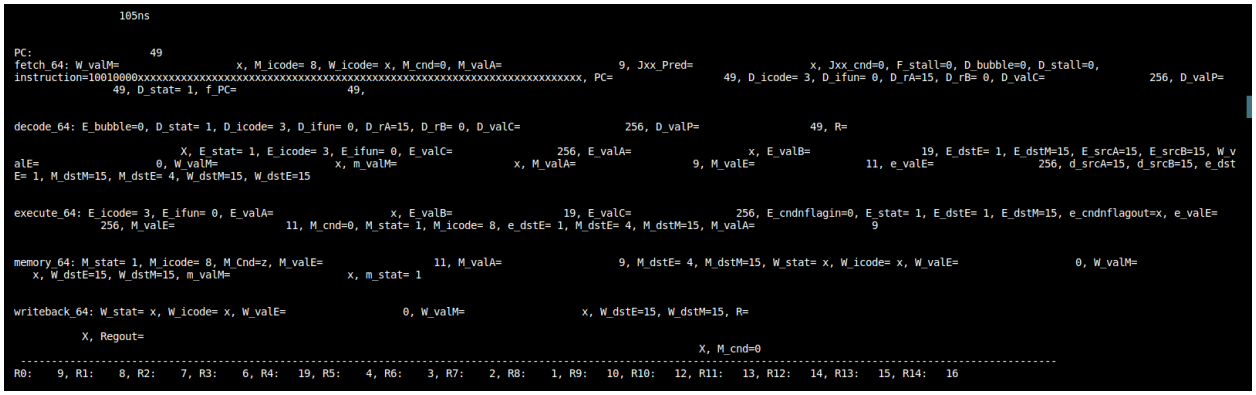
4)

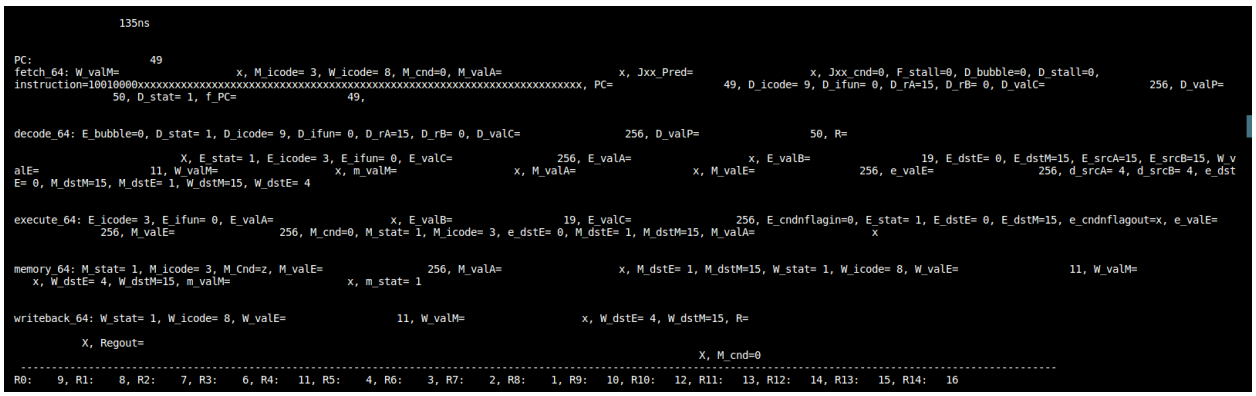
```
1   call $0x3
2   irmovq 2^6, %rcx
3   irmovq 2^6, %rax
4   irmovq $0x100, %rcx
5   irmovq $0x100, %rax
6   ret
```

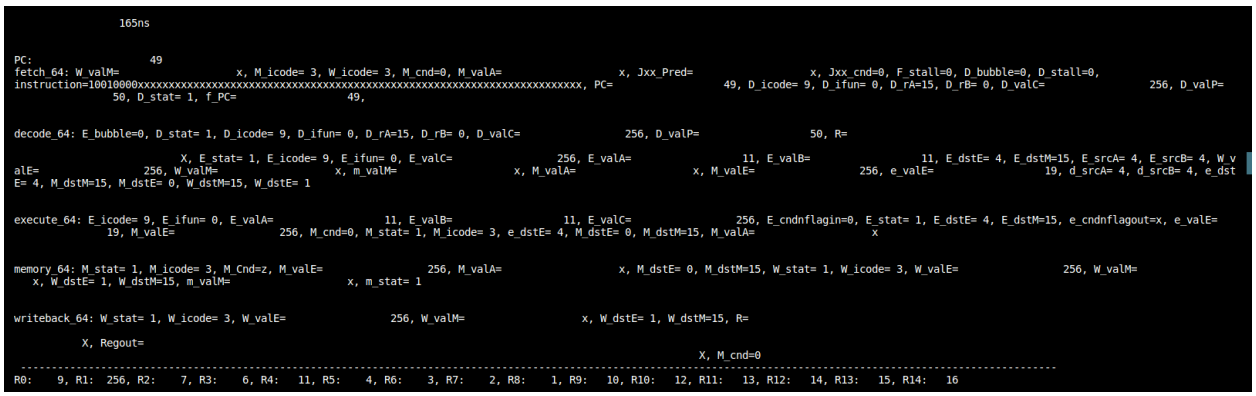
```
                  15ns

PC:                   0
fetch_64: W_valM=               x, M_icode= x, W_icode= x, M_cnd=0, M_valA=                  x, Jxx_Pred=                  x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=1000000000011101000000000000000000000000000000000000000000000000000000000000000000000110000, PC=                  0, D_icode= x, D_ifun= x, D_rA= x, D_rB= x, D_valC=                  x, D_valP=
                x, D_stat= x, f_PC=                  x,


decode_64: E_bubble=0, D_stat= x, D_icode= x, D_ifun= x, D_rA= x, D_rB= x, D_valC=                  x, D_valP=                  x, R=

                    X, E_stat= x, E_icode= x, E_ifun= x, E_valC=                  x, E_valA=                  x, E_valB=                  x, E_dstE= x, E_dstM= x, E_srcA= x, E_srcB= x, W_v
alE=                x, W_valM=                  x, m_valM=                  x, M_valA=                  x, M_valE=                  x, e_valE=                  0, d_srcA=15, d_srcB=15, e_dst
E= x, M_dstM= x, M_dstE= x, W_dstM= x, W_dstE= x


execute_64: E_icode= x, E_ifun= x, E_valA=                  x, E_valB=                  x, E_valC=                  x, E_cndnflagin=x, E_stat= x, E_dstE= x, E_dstM= x, e_cndnflagout=x, e_valE=
              0, M_valE=                  x, M_cnd=0, M_stat= x, M_icode= x, e_dstE= x, M_dstE= x, M_dstM= x, M_valA=                  x


memory_64: M_stat= x, M_icode= x, M_Cnd=z, M_valE=                  x, M_valA=                  x, M_dstE= x, M_dstM= x, W_stat= x, W_icode= x, W_valE=                  x, W_valM=
   x, W_dstE= x, W_dstM= x, m_valM=                  x, m_stat= x


writeback_64: W_stat= x, W_icode= x, W_valE=                  x, W_valM=                  x, W_dstE= x, W_dstM= x, R=

          X, Regout=

                                                                                                                                X, M_cnd=0
 ------------------------------------------------------------------------------------------------------------------------------------------------------------------------
R0:    9, R1:    8, R2:    7, R3:    6, R4:   19, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
```

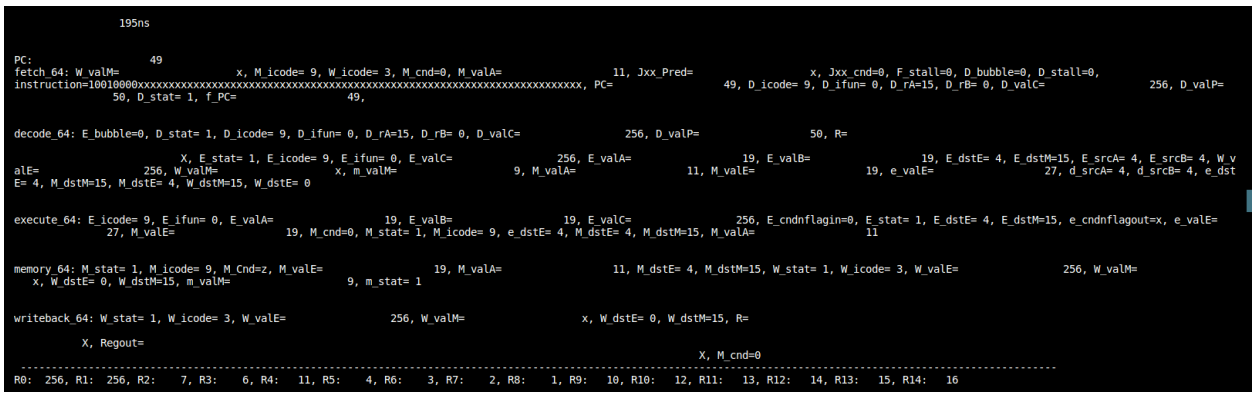
```
                  45ns

PC:                   29
fetch_64: W_valM=               x, M_icode= x, W_icode= x, M_cnd=0, M_valA=                  x, Jxx_Pred=                  x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=0011000011111000100000000000000010000000000000000000000000000000000000000000000000000000000, PC=                  29, D_icode= 8, D_ifun= 0, D_rA= x, D_rB= x, D_valC=                  29, D_valP=
                9, D_stat= 1, f_PC=                  29,


decode_64: E_bubble=0, D_stat= 1, D_icode= 8, D_ifun= 0, D_rA= x, D_rB= x, D_valC=                  29, D_valP=                  9, R=

                    X, E_stat= x, E_icode= x, E_ifun= x, E_valC=                  x, E_valA=                  9, E_valB=                  x, E_dstE=15, E_dstM=15, E_srcA=15, E_srcB=15, W_v
alE=                x, W_valM=                  x, m_valM=                  x, M_valA=                  x, M_valE=                  0, e_valE=                  0, d_srcA=15, d_srcB= 4, d_dst
E=15, M_dstM= x, M_dstE= x, W_dstM= x, W_dstE= x


execute_64: E_icode= x, E_ifun= x, E_valA=                  9, E_valB=                  x, E_valC=                  x, E_cndnflagin=x, E_stat= x, E_dstE=15, E_dstM=15, e_cndnflagout=x, e_valE=
              0, M_valE=                  0, M_cnd=0, M_stat= x, M_icode= x, e_dstE=15, M_dstE= x, M_dstM= x, M_valA=                  x


memory_64: M_stat= x, M_icode= x, M_Cnd=z, M_valE=                  0, M_valA=                  x, M_dstE= x, M_dstM= x, W_stat= x, W_icode= x, W_valE=                  x, W_valM=
   x, W_dstE= x, W_dstM= x, m_valM=                  x, m_stat= x


writeback_64: W_stat= x, W_icode= x, W_valE=                  x, W_valM=                  x, W_dstE= x, W_dstM= x, R=

          X, Regout=

                                                                                                                                X, M_cnd=0
 ------------------------------------------------------------------------------------------------------------------------------------------------------------------------
R0:    9, R1:    8, R2:    7, R3:    6, R4:   19, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
```

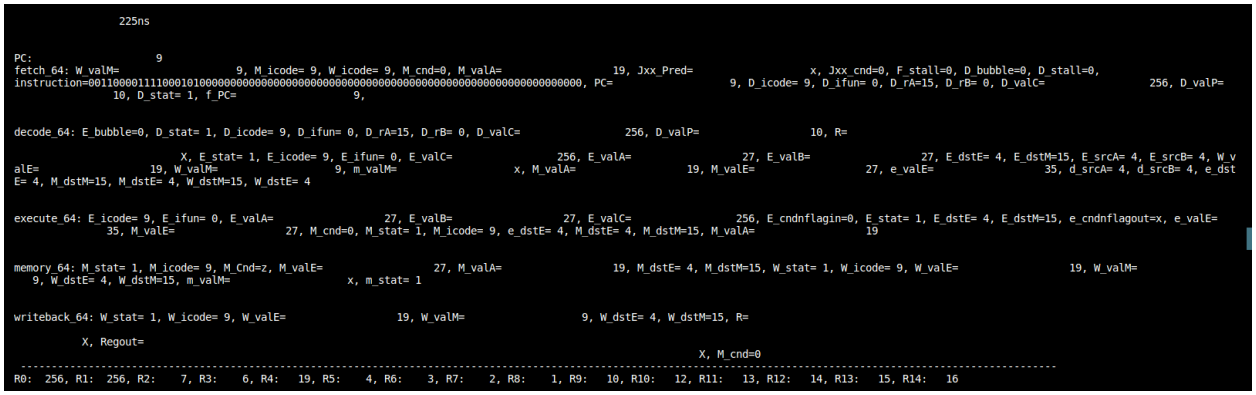
```
                  75ns

PC:                   39
fetch_64: W_valM=               x, M_icode= x, W_icode= x, M_cnd=0, M_valA=                  9, Jxx_Pred=                  x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=0011000011110000000000000000010000000000000000000000000000000000000000000000000000000000000, PC=                  39, D_icode= 3, D_ifun= 0, D_rA=15, D_rB= 1, D_valC=                256, D_valP=
               39, D_stat= 1, f_PC=                  39,


decode_64: E_bubble=0, D_stat= 1, D_icode= 3, D_ifun= 0, D_rA=15, D_rB= 1, D_valC=                256, D_valP=                39, R=

                    X, E_stat= 1, E_icode= 8, E_ifun= 0, E_valC=                 29, E_valA=                  9, E_valB=                 19, E_dstE= 4, E_dstM=15, E_srcA=15, E_srcB= 4, W_v
alE=                0, W_valM=                  x, m_valM=                  x, M_valA=                  9, M_valE=                  0, e_valE=                 11, d_srcA=15, d_srcB=15, e_dst
E= 4, M_dstM=15, M_dstE=15, W_dstM= x, W_dstE= x


execute_64: E_icode= 8, E_ifun= 0, E_valA=                  9, E_valB=                 19, E_valC=                 29, E_cndnflagin=x, E_stat= 1, E_dstE= 4, E_dstM=15, e_cndnflagout=x, e_valE=
             11, M_valE=                  0, M_cnd=0, M_stat= x, M_icode= x, e_dstE= 4, M_dstE=15, M_dstM=15, M_valA=                  9


memory_64: M_stat= x, M_icode= x, M_Cnd=z, M_valE=                  0, M_valA=                  9, M_dstE=15, M_dstM=15, W_stat= x, W_icode= x, W_valE=                  0, W_valM=
   x, W_dstE= x, W_dstM= x, m_valM=                  x, m_stat= x


writeback_64: W_stat= x, W_icode= x, W_valE=                  0, W_valM=                  x, W_dstE= x, W_dstM= x, R=

          X, Regout=

                                                                                                                                X, M_cnd=0
 ------------------------------------------------------------------------------------------------------------------------------------------------------------------------
R0:    9, R1:    8, R2:    7, R3:    6, R4:   19, R5:    4, R6:    3, R7:    2, R8:    1, R9:   10, R10:   12, R11:   13, R12:   14, R13:   15, R14:   16
```

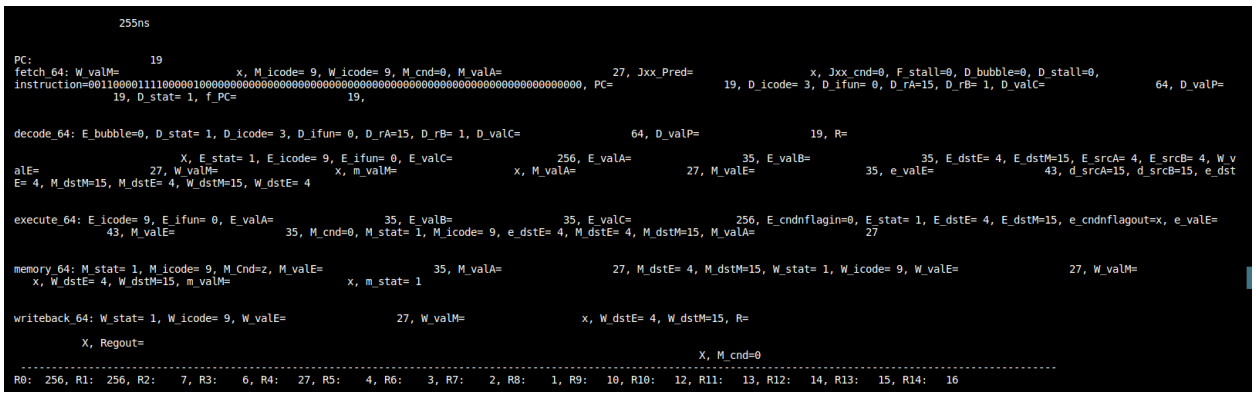

105ns
PC: 49
fetch_64: W_valM= x, M_icode= 8, W_icode= x, M_cnd=0, M_valA= 9, Jxx_Pred= x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=10010000xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, PC= 49, D_icode= 3, D_ifun= 0, D_rA=15, D_rB= 0, D_valC= 256, D_valP=
49, D_stat= 1, f_PC= 49,
decode_64: E_bubble=0, D_stat= 1, D_icode= 3, D_ifun= 0, D_rA=15, D_rB= 0, D_valC= 256, D_valP= 49, R=
X, E_stat= 1, E_icode= 3, E_ifun= 0, E_valC= 256, E_valA= x, E_valB= 19, E_dstE= 1, E_dstM=15, E_srcA=15, E_srcB=15, W_v
alE= 0, W_valM= x, m_valM= x, M_valA= 9, M_valE= 11, e_valE= 256, d_srcA=15, d_srcB=15, e_dst
E= 1, M_dstM=15, M_dstE= 4, W_dstM=15, W_dstE=15
execute_64: E_icode= 3, E_ifun= 0, E_valA= x, E_valB= 19, E_valC= 256, E_cndnflagin=0, E_stat= 1, E_dstE= 1, E_dstM=15, e_cndnflagout=x, e_valE=
256, M_valE= 11, M_cnd=0, M_stat= 1, M_icode= 8, e_dstE= 1, M_dstE= 4, M_dstM=15, M_valA= 9
memory_64: M_stat= 1, M_icode= 8, M_Cnd=z, M_valE= 11, M_valA= 9, M_dstE= 4, M_dstM=15, W_stat= x, W_icode= x, W_valE= 0, W_valM=
x, W_dstE=15, W_dstM=15, m_valM= x, m_stat= 1
writeback_64: W_stat= x, W_icode= x, W_valE= 0, W_valM= x, W_dstE=15, W_dstM=15, R=
X, Regout=
X, M_cnd=0
R0: 9, R1: 8, R2: 7, R3: 6, R4: 19, R5: 4, R6: 3, R7: 2, R8: 1, R9: 10, R10: 12, R11: 13, R12: 14, R13: 15, R14: 16

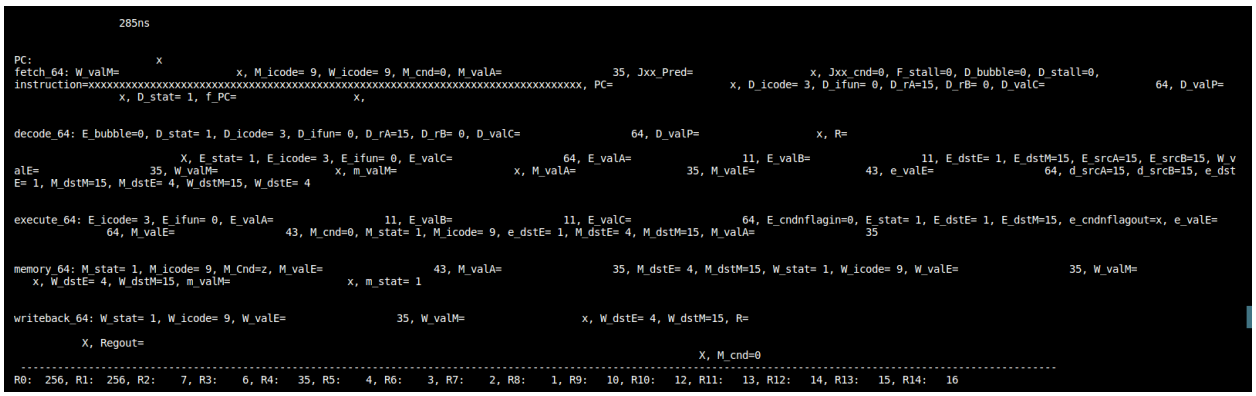

135ns
PC: 49
fetch_64: W_valM= x, M_icode= 3, W_icode= 8, M_cnd=0, M_valA= x, Jxx_Pred= x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=10010000xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, PC= 49, D_icode= 9, D_ifun= 0, D_rA=15, D_rB= 0, D_valC= 256, D_valP=
50, D_stat= 1, f_PC= 49,
decode_64: E_bubble=0, D_stat= 1, D_icode= 9, D_ifun= 0, D_rA=15, D_rB= 0, D_valC= 256, D_valP= 50, R=
X, E_stat= 1, E_icode= 3, E_ifun= 0, E_valC= 256, E_valA= x, E_valB= 19, E_dstE= 0, E_dstM=15, E_srcA=15, E_srcB=15, W_v
alE= 11, W_valM= x, m_valM= x, M_valA= x, M_valE= 256, e_valE= 256, d_srcA= 4, d_srcB= 4, e_dst
E= 0, M_dstM=15, M_dstE= 1, W_dstM=15, W_dstE= 4
execute_64: E_icode= 3, E_ifun= 0, E_valA= x, E_valB= 19, E_valC= 256, E_cndnflagin=0, E_stat= 1, E_dstE= 0, E_dstM=15, e_cndnflagout=x, e_valE=
256, M_valE= 256, M_cnd=0, M_stat= 1, M_icode= 3, e_dstE= 0, M_dstE= 1, M_dstM=15, M_valA= x
memory_64: M_stat= 1, M_icode= 3, M_Cnd=z, M_valE= 256, M_valA= x, M_dstE= 1, M_dstM=15, W_stat= 1, W_icode= 8, W_valE= 11, W_valM=
x, W_dstE= 4, W_dstM=15, m_valM= x, m_stat= 1
writeback_64: W_stat= 1, W_icode= 8, W_valE= 11, W_valM= x, W_dstE= 4, W_dstM=15, R=
X, Regout=
X, M_cnd=0
R0: 9, R1: 8, R2: 7, R3: 6, R4: 11, R5: 4, R6: 3, R7: 2, R8: 1, R9: 10, R10: 12, R11: 13, R12: 14, R13: 15, R14: 16

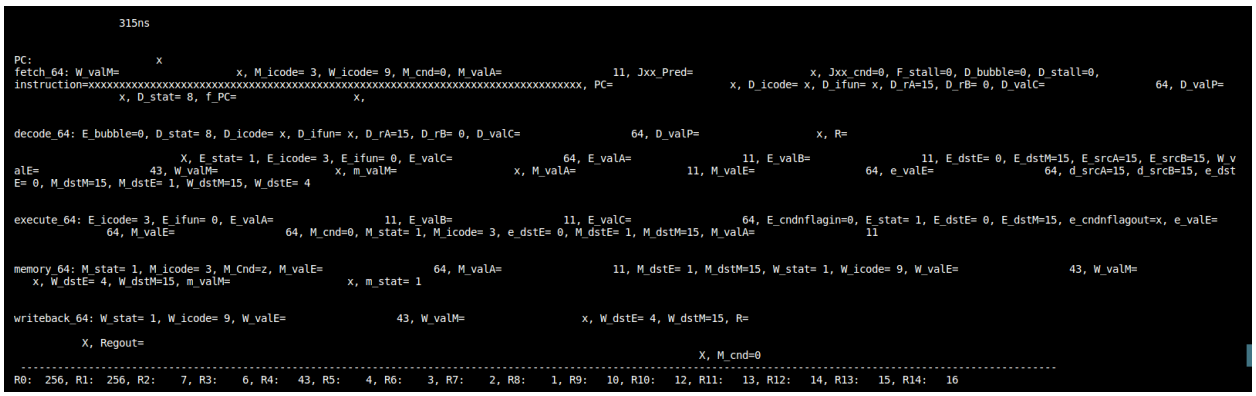

165ns
PC: 49
fetch_64: W_valM= x, M_icode= 3, W_icode= 3, M_cnd=0, M_valA= x, Jxx_Pred= x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=10010000xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, PC= 49, D_icode= 9, D_ifun= 0, D_rA=15, D_rB= 0, D_valC= 256, D_valP=
50, D_stat= 1, f_PC= 49,
decode_64: E_bubble=0, D_stat= 1, D_icode= 9, D_ifun= 0, D_rA=15, D_rB= 0, D_valC= 256, D_valP= 50, R=
X, E_stat= 1, E_icode= 9, E_ifun= 0, E_valC= 256, E_valA= 11, E_valB= 11, E_dstE= 4, E_dstM=15, E_srcA= 4, E_srcB= 4, W_v
alE= 256, W_valM= x, m_valM= x, M_valA= x, M_valE= 256, e_valE= 19, d_srcA= 4, d_srcB= 4, e_dst
E= 4, M_dstM=15, M_dstE= 0, W_dstM=15, W_dstE= 1
execute_64: E_icode= 9, E_ifun= 0, E_valA= 11, E_valB= 11, E_valC= 256, E_cndnflagin=0, E_stat= 1, E_dstE= 4, E_dstM=15, e_cndnflagout=x, e_valE=
19, M_valE= 256, M_cnd=0, M_stat= 1, M_icode= 3, e_dstE= 4, M_dstE= 0, M_dstM=15, M_valA= x
memory_64: M_stat= 1, M_icode= 3, M_Cnd=z, M_valE= 256, M_valA= x, M_dstE= 0, M_dstM=15, W_stat= 1, W_icode= 3, W_valE= 256, W_valM=
x, W_dstE= 1, W_dstM=15, m_valM= x, m_stat= 1
writeback_64: W_stat= 1, W_icode= 3, W_valE= 256, W_valM= x, W_dstE= 1, W_dstM=15, R=
X, Regout=
X, M_cnd=0
R0: 9, R1: 256, R2: 7, R3: 6, R4: 11, R5: 4, R6: 3, R7: 2, R8: 1, R9: 10, R10: 12, R11: 13, R12: 14, R13: 15, R14: 16

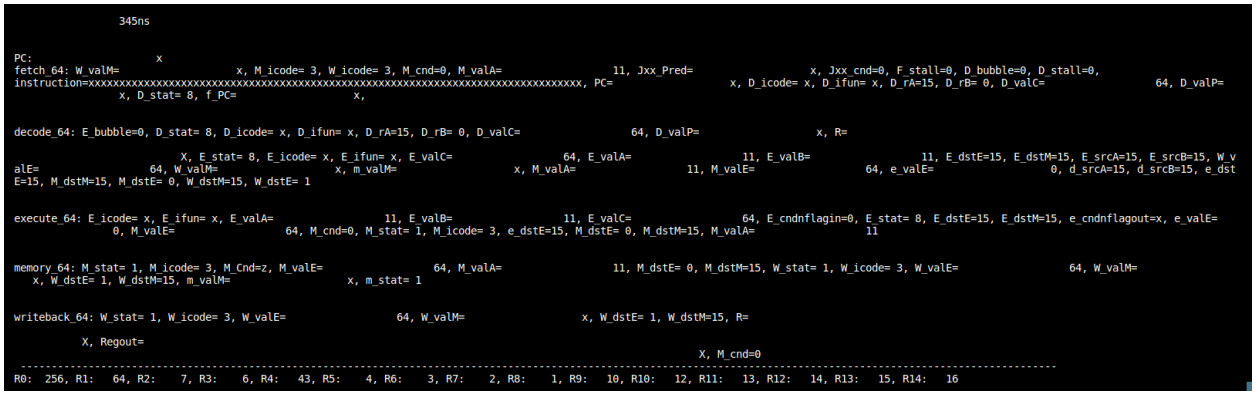

195ns
PC: 49
fetch_64: W_valM= x, M_icode= 9, W_icode= 3, M_cnd=0, M_valA= 11, Jxx_Pred= x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=10010000xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, PC= 49, D_icode= 9, D_ifun= 0, D_rA=15, D_rB= 0, D_valC= 256, D_valP=
50, D_stat= 1, f_PC= 49,
decode_64: E_bubble=0, D_stat= 1, D_icode= 9, D_ifun= 0, D_rA=15, D_rB= 0, D_valC= 256, D_valP= 50, R=
X, E_stat= 1, E_icode= 9, E_ifun= 0, E_valC= 256, E_valA= 19, E_valB= 19, E_dstE= 4, E_dstM=15, E_srcA= 4, E_srcB= 4, W_v
alE= 256, W_valM= x, m_valM= 9, M_valA= 11, M_valE= 19, e_valE= 27, d_srcA= 4, d_srcB= 4, e_dst
E= 4, M_dstM=15, M_dstE= 4, W_dstM=15, W_dstE= 0
execute_64: E_icode= 9, E_ifun= 0, E_valA= 19, E_valB= 19, E_valC= 256, E_cndnflagin=0, E_stat= 1, E_dstE= 4, E_dstM=15, e_cndnflagout=x, e_valE=
27, M_valE= 19, M_cnd=0, M_stat= 1, M_icode= 9, e_dstE= 4, M_dstE= 4, M_dstM=15, M_valA= 11
memory_64: M_stat= 1, M_icode= 9, M_Cnd=z, M_valE= 19, M_valA= 11, M_dstE= 4, M_dstM=15, W_stat= 1, W_icode= 3, W_valE= 256, W_valM=
x, W_dstE= 0, W_dstM=15, m_valM= 9, m_stat= 1
writeback_64: W_stat= 1, W_icode= 3, W_valE= 256, W_valM= x, W_dstE= 0, W_dstM=15, R=
X, Regout=
X, M_cnd=0
R0: 256, R1: 256, R2: 7, R3: 6, R4: 11, R5: 4, R6: 3, R7: 2, R8: 1, R9: 10, R10: 12, R11: 13, R12: 14, R13: 15, R14: 16

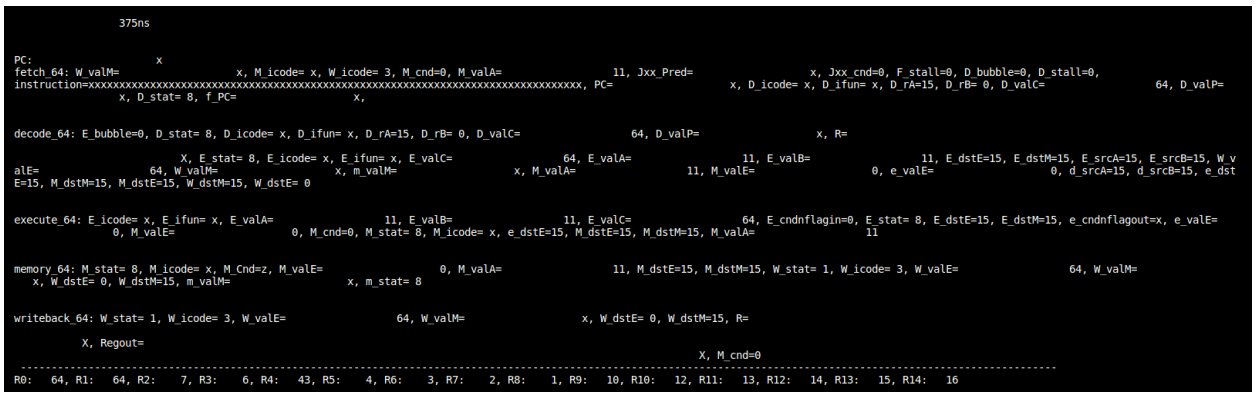

225ns
PC: 9
fetch_64: W_valM= 9, M_icode= 9, W_icode= 9, M_cnd=0, M_valA= 19, Jxx_Pred= x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=0011000011110001010000000000000000000000000000000000000000000000000000000000000000, PC= 9, D_icode= 9, D_ifun= 0, D_rA=15, D_rB= 0, D_valC= 256, D_valP=
10, D_stat= 1, f_PC= 9,
decode_64: E_bubble=0, D_stat= 1, D_icode= 9, D_ifun= 0, D_rA=15, D_rB= 0, D_valC= 256, D_valP= 10, R=
X, E_stat= 1, E_icode= 9, E_ifun= 0, E_valC= 256, E_valA= 27, E_valB= 27, E_dstE= 4, E_dstM=15, E_srcA= 4, E_srcB= 4, W_v
alE= 19, W_valM= 9, m_valM= x, M_valA= 19, M_valE= 27, e_valE= 35, d_srcA= 4, d_srcB= 4, e_dst
E= 4, M_dstM=15, M_dstE= 4, W_dstM=15, W_dstE= 4
execute_64: E_icode= 9, E_ifun= 0, E_valA= 27, E_valB= 27, E_valC= 256, E_cndnflagin=0, E_stat= 1, E_dstE= 4, E_dstM=15, e_cndnflagout=x, e_valE=
35, M_valE= 27, M_cnd=0, M_stat= 1, M_icode= 9, e_dstE= 4, M_dstE= 4, M_dstM=15, M_valA= 19
memory_64: M_stat= 1, M_icode= 9, M_Cnd=z, M_valE= 27, M_valA= 19, M_dstE= 4, M_dstM=15, W_stat= 1, W_icode= 9, W_valE= 19, W_valM=
9, W_dstE= 4, W_dstM=15, m_valM= x, m_stat= 1
writeback_64: W_stat= 1, W_icode= 9, W_valE= 19, W_valM= 9, W_dstE= 4, W_dstM=15, R=
X, Regout=
X, M_cnd=0
R0: 256, R1: 256, R2: 7, R3: 6, R4: 19, R5: 4, R6: 3, R7: 2, R8: 1, R9: 10, R10: 12, R11: 13, R12: 14, R13: 15, R14: 16


255ns
PC: 19
fetch_64: W_valM= x, M_icode= 9, W_icode= 9, M_cnd=0, M_valA= 27, Jxx_Pred= x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=0011000011110000100000000000000000000000000000000000000000000000000000000000000000, PC= 19, D_icode= 3, D_ifun= 0, D_rA=15, D_rB= 1, D_valC= 64, D_valP=
19, D_stat= 1, f_PC= 19,
decode_64: E_bubble=0, D_stat= 1, D_icode= 3, D_ifun= 0, D_rA=15, D_rB= 1, D_valC= 64, D_valP= 19, R=
X, E_stat= 1, E_icode= 9, E_ifun= 0, E_valC= 256, E_valA= 35, E_valB= 35, E_dstE= 4, E_dstM=15, E_srcA= 4, E_srcB= 4, W_v
alE= 27, W_valM= x, m_valM= x, M_valA= 27, M_valE= 35, e_valE= 43, d_srcA=15, d_srcB=15, e_dst
E= 4, M_dstM=15, M_dstE= 4, W_dstM=15, W_dstE= 4
execute_64: E_icode= 9, E_ifun= 0, E_valA= 35, E_valB= 35, E_valC= 256, E_cndnflagin=0, E_stat= 1, E_dstE= 4, E_dstM=15, e_cndnflagout=x, e_valE=
43, M_valE= 35, M_cnd=0, M_stat= 1, M_icode= 9, e_dstE= 4, M_dstE= 4, M_dstM=15, M_valA= 27
memory_64: M_stat= 1, M_icode= 9, M_Cnd=z, M_valE= 35, M_valA= 27, M_dstE= 4, M_dstM=15, W_stat= 1, W_icode= 9, W_valE= 27, W_valM=
x, W_dstE= 4, W_dstM=15, m_valM= x, m_stat= 1
writeback_64: W_stat= 1, W_icode= 9, W_valE= 27, W_valM= x, W_dstE= 4, W_dstM=15, R=
X, Regout=
X, M_cnd=0
R0: 256, R1: 256, R2: 7, R3: 6, R4: 27, R5: 4, R6: 3, R7: 2, R8: 1, R9: 10, R10: 12, R11: 13, R12: 14, R13: 15, R14: 16


285ns
PC: x
fetch_64: W_valM= x, M_icode= 9, W_icode= 9, M_cnd=0, M_valA= 35, Jxx_Pred= x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, PC= x, D_icode= 3, D_ifun= 0, D_rA=15, D_rB= 0, D_valC= 64, D_valP=
x, D_stat= 1, f_PC= x,
decode_64: E_bubble=0, D_stat= 1, D_icode= 3, D_ifun= 0, D_rA=15, D_rB= 0, D_valC= 64, D_valP= x, R=
X, E_stat= 1, E_icode= 3, E_ifun= 0, E_valC= 64, E_valA= 11, E_valB= 11, E_dstE= 1, E_dstM=15, E_srcA=15, E_srcB=15, W_v
alE= 35, W_valM= x, m_valM= x, M_valA= 35, M_valE= 43, e_valE= 64, d_srcA=15, d_srcB=15, e_dst
E= 1, M_dstM=15, M_dstE= 4, W_dstM=15, W_dstE= 4
execute_64: E_icode= 3, E_ifun= 0, E_valA= 11, E_valB= 11, E_valC= 64, E_cndnflagin=0, E_stat= 1, E_dstE= 1, E_dstM=15, e_cndnflagout=x, e_valE=
64, M_valE= 43, M_cnd=0, M_stat= 1, M_icode= 9, e_dstE= 1, M_dstE= 4, M_dstM=15, M_valA= 35
memory_64: M_stat= 1, M_icode= 9, M_Cnd=z, M_valE= 43, M_valA= 35, M_dstE= 4, M_dstM=15, W_stat= 1, W_icode= 9, W_valE= 35, W_valM=
x, W_dstE= 4, W_dstM=15, m_valM= x, m_stat= 1
writeback_64: W_stat= 1, W_icode= 9, W_valE= 35, W_valM= x, W_dstE= 4, W_dstM=15, R=
X, Regout=
X, M_cnd=0
R0: 256, R1: 256, R2: 7, R3: 6, R4: 35, R5: 4, R6: 3, R7: 2, R8: 1, R9: 10, R10: 12, R11: 13, R12: 14, R13: 15, R14: 16


315ns
PC: x
fetch_64: W_valM= x, M_icode= 3, W_icode= 9, M_cnd=0, M_valA= 11, Jxx_Pred= x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, PC= x, D_icode= x, D_ifun= x, D_rA=15, D_rB= 0, D_valC= 64, D_valP=
x, D_stat= 8, f_PC= x,
decode_64: E_bubble=0, D_stat= 8, D_icode= x, D_ifun= x, D_rA=15, D_rB= 0, D_valC= 64, D_valP= x, R=
X, E_stat= 1, E_icode= 3, E_ifun= 0, E_valC= 64, E_valA= 11, E_valB= 11, E_dstE= 0, E_dstM=15, E_srcA=15, E_srcB=15, W_v
alE= 43, W_valM= x, m_valM= x, M_valA= 11, M_valE= 64, e_valE= 64, d_srcA=15, d_srcB=15, e_dst
E= 0, M_dstM=15, M_dstE= 1, W_dstM=15, W_dstE= 4
execute_64: E_icode= 3, E_ifun= 0, E_valA= 11, E_valB= 11, E_valC= 64, E_cndnflagin=0, E_stat= 1, E_dstE= 0, E_dstM=15, e_cndnflagout=x, e_valE=
64, M_valE= 64, M_cnd=0, M_stat= 1, M_icode= 3, e_dstE= 0, M_dstE= 1, M_dstM=15, M_valA= 11
memory_64: M_stat= 1, M_icode= 3, M_Cnd=z, M_valE= 64, M_valA= 11, M_dstE= 1, M_dstM=15, W_stat= 1, W_icode= 9, W_valE= 43, W_valM=
x, W_dstE= 4, W_dstM=15, m_valM= x, m_stat= 1
writeback_64: W_stat= 1, W_icode= 9, W_valE= 43, W_valM= x, W_dstE= 4, W_dstM=15, R=
X, Regout=
X, M_cnd=0
R0: 256, R1: 256, R2: 7, R3: 6, R4: 43, R5: 4, R6: 3, R7: 2, R8: 1, R9: 10, R10: 12, R11: 13, R12: 14, R13: 15, R14: 16


345ns
PC: x
fetch_64: W_valM= x, M_icode= 3, W_icode= 3, M_cnd=0, M_valA= 11, Jxx_Pred= x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, PC= x, D_icode= x, D_ifun= x, D_rA=15, D_rB= 0, D_valC= 64, D_valP=
x, D_stat= 8, f_PC= x,
decode_64: E_bubble=0, D_stat= 8, D_icode= x, D_ifun= x, D_rA=15, D_rB= 0, D_valC= 64, D_valP= x, R=
X, E_stat= 8, E_icode= x, E_ifun= x, E_valC= 64, E_valA= 11, E_valB= 11, E_dstE=15, E_dstM=15, E_srcA=15, E_srcB=15, W_v
alE= 64, W_valM= x, m_valM= x, M_valA= 11, M_valE= 64, e_valE= 0, d_srcA=15, d_srcB=15, e_dst
E=15, M_dstM=15, M_dstE= 0, W_dstM=15, W_dstE= 1
execute_64: E_icode= x, E_ifun= x, E_valA= 11, E_valB= 11, E_valC= 64, E_cndnflagin=0, E_stat= 8, E_dstE=15, E_dstM=15, e_cndnflagout=x, e_valE=
0, M_valE= 64, M_cnd=0, M_stat= 1, M_icode= 3, e_dstE=15, M_dstE= 0, M_dstM=15, M_valA= 11
memory_64: M_stat= 1, M_icode= 3, M_Cnd=z, M_valE= 64, M_valA= 11, M_dstE= 0, M_dstM=15, W_stat= 1, W_icode= 3, W_valE= 64, W_valM=
x, W_dstE= 1, W_dstM=15, m_valM= x, m_stat= 1
writeback_64: W_stat= 1, W_icode= 3, W_valE= 64, W_valM= x, W_dstE= 1, W_dstM=15, R=
X, Regout=
X, M_cnd=0
R0: 256, R1: 64, R2: 7, R3: 6, R4: 43, R5: 4, R6: 3, R7: 2, R8: 1, R9: 10, R10: 12, R11: 13, R12: 14, R13: 15, R14: 16


375ns
PC: x
fetch_64: W_valM= x, M_icode= x, W_icode= 3, M_cnd=0, M_valA= 11, Jxx_Pred= x, Jxx_cnd=0, F_stall=0, D_bubble=0, D_stall=0,
instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, PC= x, D_icode= x, D_ifun= x, D_rA=15, D_rB= 0, D_valC= 64, D_valP=
x, D_stat= 8, f_PC= x,
decode_64: E_bubble=0, D_stat= 8, D_icode= x, D_ifun= x, D_rA=15, D_rB= 0, D_valC= 64, D_valP= x, R=
X, E_stat= 8, E_icode= x, E_ifun= x, E_valC= 64, E_valA= 11, E_valB= 11, E_dstE=15, E_dstM=15, E_srcA=15, E_srcB=15, W_v
alE= 64, W_valM= x, m_valM= x, M_valA= 11, M_valE= 0, e_valE= 0, d_srcA=15, d_srcB=15, e_dst
E=15, M_dstM=15, M_dstE=15, W_dstM=15, W_dstE= 0
execute_64: E_icode= x, E_ifun= x, E_valA= 11, E_valB= 11, E_valC= 64, E_cndnflagin=0, E_stat= 8, E_dstE=15, E_dstM=15, e_cndnflagout=x, e_valE=
0, M_valE= 0, M_cnd=0, M_stat= 8, M_icode= x, e_dstE=15, M_dstE=15, M_dstM=15, M_valA= 11
memory_64: M_stat= 8, M_icode= x, M_Cnd=z, M_valE= 0, M_valA= 11, M_dstE=15, M_dstM=15, W_stat= 1, W_icode= 3, W_valE= 64, W_valM=
x, W_dstE= 0, W_dstM=15, m_valM= x, m_stat= 8
writeback_64: W_stat= 1, W_icode= 3, W_valE= 64, W_valM= x, W_dstE= 0, W_dstM=15, R=
X, Regout=
X, M_cnd=0
R0: 64, R1: 64, R2: 7, R3: 6, R4: 43, R5: 4, R6: 3, R7: 2, R8: 1, R9: 10, R10: 12, R11: 13, R12: 14, R13: 15, R14: 16