Homework 10

Problem 1: SEQ Implementation

Assume we are going to implementation the following instruction in our SEQ y68_64 processor: iaddq V, rB, which added value V to the value stored in register rB (R[rB] <- R[rB] + V);

1. Try to fill in the table listed below.

Stage	iaddq V, rB
Fetch	icode:ifun <- M1[PC] rA:rB <- M1[PC+1] valC <- M8[PC+2] valP <- PC+10
Decode	valB <- R[rB]
Execute	valE <- valB + valC
Memory	
Write back	R[rB] <- valE
PC update	PC <- valP

2. Please point out which logic blocks in the following ones should be modified and write the modified HCL code below.

{need_valC, srcB, dstM, aluB, mem_data}

```
bool need_valC = icode in { IIRMOVQ, IRMMOVQ, IMRMOVQ, IJXX, ICALL, IIADDQ}

int srcB = [
    icode in { IOPQ, IRMMOVQ, IMRMOVQ, IIADDQ } : rB;
    icode in { IPUSHQ, IPOPQ, ICALL, IRET } : RESP;
    1 : RNONE;   # Don’t need register
];

int aluB = [
    icode in { IRMMOVQ, IMRMOVQ, IOPQ, ICALL, IPUSHQ, IRET, IPOPQ,
IIADDQ } : valB;
    icode in { IRRMOVL, IIRMOVL} : 0
};
```

Problem 2: Pipeline

Assume now we have 4 load of clothes need to wash, dry, and fold. The wash task takes 40 minutes, dry task takes 50 minutes, and fold task takes 30 minutes to deal with one load of clothes.

If we do these tasks sequentially, how long would the whole process take? What if we have more resources and do these tasks using Pipeline? Please indicate the Speedup (old/new).

SEQ: (40+50+30) *4 = 480min = 8h
PIPE: 40 + 50 * 4 + 30 = 270min = 4.5h
Speedup: 8/4.8 = 1.78

Problem 3: Data hazard

Is there a data hazard in the following codes in our **PIPELINE** processor **WITHOUT** forwarding? If yes, explain how the hazard happens, insert nops to solve the hazard and show what value should `d_valA` and `d_valB` take if we solve this problem by forwarding. If no, explain why.

(**NOTE**: you should represent `d_valA` and `d_valB` by other HCL variables like `E_dstE`)

p1:	p2:
1 irmovq $1, %rbx 2 irmovq $2, %rcx 3 addl %rbx, %rcx	1 rrmovq %rax, %rbx 2 irmovq $1, %rax

P1: there is a data hazard, the decode stage of line 3 should happen after the write back stage of line 1 and line 2 in order to get the newest value of %rbx and %rcx.
The data hazard in P1 can be solved by insert 3 nops like following:

```
p1:
1 irmovq $1, %rbx
2 irmovq $2, %rcx
4 nop
5 nop
6 nop
7 addl %rbx, %rcx
```

When using forwarding, let `d_valA=M_valE` and `d_valB=e_valE`.

P2: No data hazard, value of %rax hasn't been modified by line 2 at the decode stage of line 1.