VE370 Homework4

Q1

`lw $t0, -4($sp)`

> As the MUX used to choose between data memory and ALU result do not specify ports, here assume 1 as choose from data memory.

1. IF: in PC: the addresss of instruction `lw $t0, -4($sp)`
2. ID: in IF/ID: machine code of instruction `lw $t0, -4($sp)` , 010111 $sp 01000 1111 1111 1111 1100 and $PC+4$.
3. EX: in ID/EX:

WB	*M*	*EX*	*Reg.File out*	
RegWrite: 1	MemRead: 1	RegDst: X, set as 0	Read Data1: R[$sp]	IF/ID.RegisterRs: $sp
MemtoReg:0	MemWrite: 0	ALUSrc: 1	Read Data2: R[$t0]	IF/ID.RegisterRt: $t0
	Branch: 0	ALUOp: 00		IF/ID.RegisterRd: 111 11
				$PC+4$
				$offset(-4)$

4. MEM: in EX/MEM:

WB	*M*	*ALU out*	
RegWrite: 1	MemRead: 1	Zero: not know	Read Data2: R[$t0], as Write Data
MemtoReg:0	MemWrite: 0	Result: R[$sp]-4	$t0, write Register Dst
	Branch: 0		$PC+4+offset(-4)$

5. WB: in MEM/WB:

WB	***Data Mem out***	
RegWrite: 1	Mem[R[$sp]-4]	ALU Result: R[$sp]-4
MemtoReg:0		$t0, write Register Dst

Q2

```
L1: sw $18,–12($8)
L2: lw $3,8($18)
L3: add $6,$3,$3
L4: or $8,$9,$6
```

1. register $3 between instruction L2 and L3

   register $6 between instruction L3 and L4
2. hazard same as in 1. add 2 NOP between instruction L2 and L3, then add 2 NOP between instruction L3 and L4. 12 clock cycles.
3. hazard: register $3 between instruction L2 and L3

   add two NOP between L2 and L3. 10 clock cycles.
4. hazard: register $3 between instruction L2 and L3

   add one NOP between L2 and L3. 9 clock cycles.

Q3

```
    lw  $t0, Offs(Rs)
    bne Rt, $t0, skip
    jr  Rd
skip: ...
```

2. need **add a read register in Reg File**, because read three register (`Rd`, `Rt`, `Rs`). For `Rt` and `Rs` they could be read through the previous read port as the operation just like `lw temp, offset(Rs)`. Then after the Mem stage, `temp=Mem[Offs+Rs]`, **need to compare the value of temp and Rt**. Then, to change the value of PC **need to add R[Rd] as the input of PC MUX and a select signal that has the result of comparison**.

3. a) Control signal `BEQM` test whether we have this instruction and the comparison result. Like Branch, it has an and GATE with another input as the comparison result of `Mem[Offs+Rs]` and `Rt`.

   b) Control signal to select whether the PC should be `R[Rd]`

4. Data hazard could be resolved by adding a forwarding path, from new stage to EX.

   Control hazard created by new PC is unknown until `beqm` is finished. It would be longer as the result is unknown until MEM and comparison is done.

Q4

```
L1: sub $6, $2, $1
L2: lw  $3, 8($6)
L3: lw  $2, 0($6)
L4: or  $3, $5, $3
L5: sw  $3, 0($5)
```

1.

	1	2	3	4	5
L1	IF	ID	EX($6=$2-$1)	MEM	WB
L2		IF	ID($6)	EX	MEM($3=MEM[8+$6])
L3			IF	ID($6)	EX
L4				IF	ID
L5					IF

without hazard detection unit, all will be correct, as no instruction need the register that just be loaded through the `load` instruction. (L3 do not need $3, L4 do not need $2)

2.

	PC Write	IF/ID Write	Hazard	ID/EX Mem Read	IF/ID Rs	IF/ID Rt	ID/EX Rt	ID/EX Rs	EX/MEM Dst	EX/MEM RegWrite	ForwardA	Forward B	MEM/WB Dst	MEM/WB RegWrite
CC1	1	1	x	x	x	x	x	x	x	x	x	x	x	x
CC2	1	1	x	x	$2	$1	x	x	x	x	x	x	x	x
CC3	1	1	0	0	$6	$3	$1	$2	x	x	00	00	x	x
CC4	1	1	0	1	$6	$2	$3	$6	$6	1	10	00	x	x
CC5	1	1	0	1	$3	$5	$2	$6	$3	1	00	00	$6	0

3.

- Check destination register of ID/EX and Regwrite with IF/ID.Rt/Rs, need to add stall when dependence (register $6 between L1 and L2, register $3 between L4 and L5)
- Check destination register of EX/MEM and Regwrite with IF/ID.Rt/Rs, need to add stall when dependence (register $6 between L1 and L3, register $3 between L2 and L4)