VE370 HW4 Solution

1.

a. PC: PC+4

b. IF/ID: 100011 11101 01000 1111 1111 1111 1100 (0x8FA8FFFC)

c. ID/EX: WB(RegWrite)=1
M(MemWrite)=0
M(MemRead)=1
ReadData1=[$sp]
ReadData2=[$t0]
IF/ID.RegisterRs=11101
IF/ID.RegisterRt=01000
IF/ID.RegisterRd=11111

d. EX/MEM: WB(RegWrite)=1
M(MemWrite)=0
M(MemRead)=1
ALU Result=[$sp]-4
ReadData2=[$t0]
Rd=01000

e. MEM/WB: WB(RegWrite)=1
Mem ReadData=Mem[[$sp]-4]
ALU Result=[$sp]-4
Rd=01000

2.

(a) $3 between L2 and L3.
$6 between L3 and L4.

(b)

```
sw $18, -12($8)
lw $3, 8($18)
nop
nop
add $6, $3, $3
nop
nop
or $8, $9, $6
```

12 clock cycles

(c)

```
sw $18, -12($8)
lw $3, 8($18)
nop
nop
add $6, $3, $3
or $8, $9, $6
```

10 clock cycles

(d)

```
sw $18, -12($8)
lw $3, 8($18)
nop
add $6, $3, $3
or $8, $9, $6
```

9 clock cycles

3.

(a)

```
lw Rtmp, Offs(Rs)
bne Rtmp, Rt, Skip
jr Rd
Skip: ...
```

(b)

1. An extra comparator added to the MEM or WB stage (or a new stage) so that we can compare the value read from memory and Rt.
2. Add an input to the PC Mux that takes the value of Rd, and the Mux select signal must include the result of the new comparison.
3. An extra read port in Register File because the instruction needs three registers to be read.

(c)

1. A control signal similar to the existing "Branch" signal to control whether or not the new comparison is allowed to affect the PC.
2. An extra bit to the control signal that selects whether the target address is PC+4+Offs or the register value.

(d)

It depends on your design in (b) and (c). For example, if an extra comparator is added in the MEM stage, then the new instruction creates a new kind of control hazard that leaves the next PC unknown until this new instruction leaves the MEM stage, which is one cycle longer than for a normal BEQ. Meanwhile, forwarding unit may be modified based on your design.

Note: The solution in (b), (c) and (d) is not unique. Feel free to discuss with TA.

4.

(a) The code can be executed correctly.

(b)

Cycle #1: PCWrite=1, ForwardA=X, ForwardB=X
Cycle #2: PCWrite=1, ForwardA=X, ForwardB=X
Cycle #3: PCWrite=1, ForwardA=00, ForwardB=00
Cycle #4: PCWrite=1, ForwardA=10, ForwardB=00
Cycle #5: PCWrite=1, ForwardA=01, ForwardB=00

(c)

Without forwarding path, we need to detect all hazards and insert stalls.

the existing hazard detection unit looks at these signals:
ID/EX.MemRead and
  ((ID/EX.RegisterRt == IF/ID.RegisterRs) or
   (ID/EX.RegisterRt == IF/ID.RegisterRt))

To resolve hazard between sub $6 and lw $3,8($6) as well as lw $2,0($6), we need to stall lw for two clock cycles, so we need to know
1. there is a load instruction in IF/ID: IF/ID.MemRead = 1, IF/ID.RegisterRs
2. there is an R type in ID/EX and data dependency: ID/EX.RegWrite and ID/EX.RegisterRd
or
3. there is an R type in EX/MEM and data dependency: EX/MEM.RegWrite and EX/MEM.RegisterRd

To resolve hazard between lw $3,8($6) and or $3, $5, $3, we need to stall the OR by one clock cycle, so we need to know:
1. there is a lw in EX/MEM: EX/MEM.MemRead = 1, EX/MEM.RegisterRt
2. data dependency: IF/ID.RegWrite and IF/ID.RegisterRs/Rt

No extra output is needed.