4.1

AND Rd, Rs, Rt

4.1.1

RegWrite	1
ALU mux	0
ALU OP	AND
MEM write	0
MEm Read	0
Branch	0
Reg MUX	1

for MUX I assumed 

4.1.2 All except Data Memory

4.1.3 Adder PC+4 + Im

4.2

LWI $R_t, R_d(R_s)$

$Reg[R_t] : M[Reg[R_d] + Reg[R_s]]$

for the effective address we need to add 2 Registers instead of a Reg + Imm. we do have this capability already

4.2.1 Use the ALU + Mem

4.2.2 No need

4.2.3 we can use the existing signals just set ALU mux to 0 instead of 1

4.4

4.4.1 we need to fetch inst and set PC ← PC + 4

Mem takes longer → 200 ps

4.4.2

critical Path

I-Mem + sign extend + shift left + Add + Mux

200 + 15 + 10 + 70 + 20

315 ps.

4.4.3 The new path

IM + Reg read + MUX + ALU + MUX

200 + 90 + 20 + 90 + 20 ∴ 420 ps

420 > 315 The new path: 420 ps

4.4.4 PC relative branch

4.4.5 PC relative unconditional since conditional goes through ALU

Which is longer

4.4.6 Between beq & Add

beq takes more (both use ALU but beq goes through Add)

To have an effect we have to change ~~3~~ 4.4.2 to 4.4.3

i.e. 315 → 420

we have to increase it by

105 ps

4.9

```
or    r1, r2, r3
or    r2, r1, r4
or    r1, r1, r2
```

4.9.1 see the arrows above (we did not cover the rest)

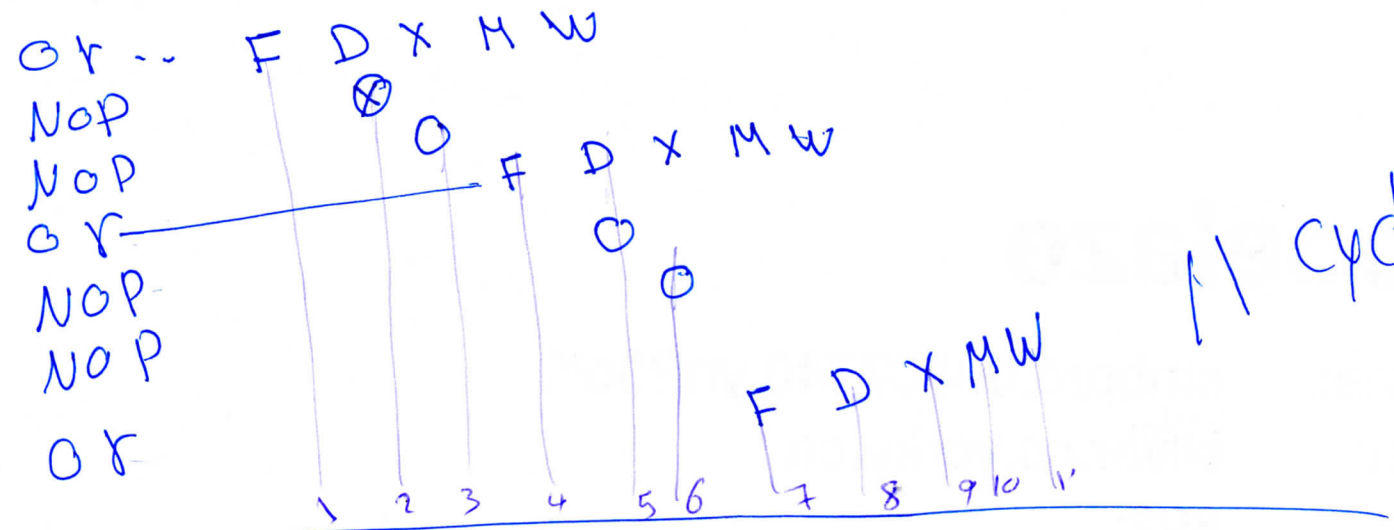
4.9.2

```
or    r1, r2, r3
nop
nop
or    r2, r1, r4
NOP
NOP
or    r1, r1, r2
```

4.9.3 same as 4.9 no NOP

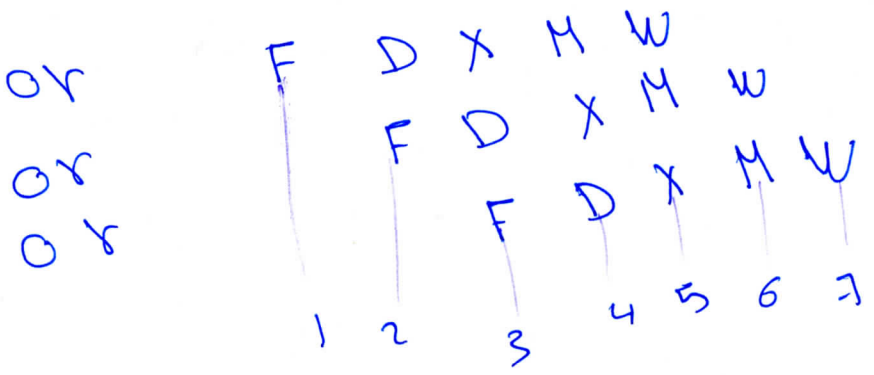
4.9.4

without forwarding



11 cycles

with forwarding



7 cycles

multiply by cycle time