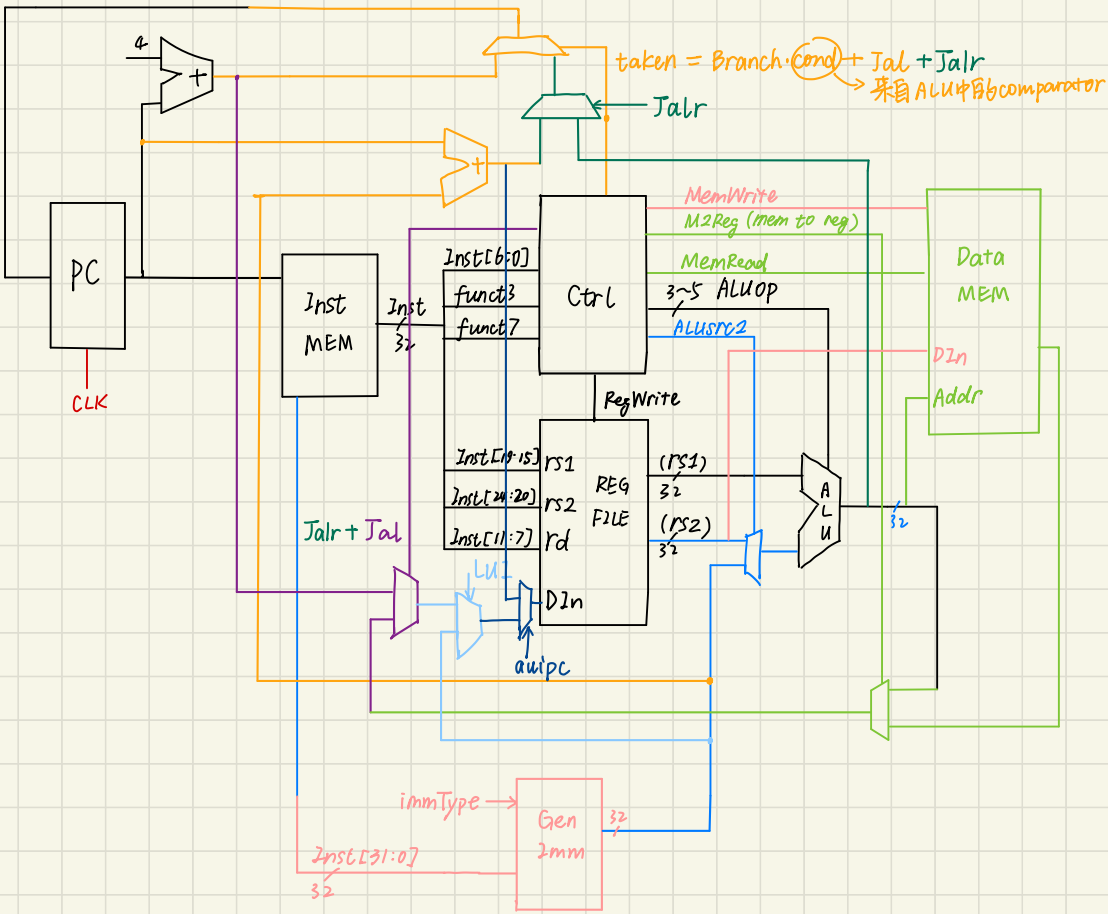
HW4

4.7 说明：① 各指令的 latency 计算都基于完整的数据通路

② 如有 n 个 2 选 1 的 mux，只计算一次 Mux 的时延 (25ps)

因为题中未指明 Mux 是何种多路选择器



4.7.1 R-type

Register Read (for PC) + I-Mem + RegFile + MUX + ALU + MUX

+ Register Setup (for RegFile)

= (30 + 250 + 150 + 25 + 200 + 25 + 20) ps = 700 ps

4.7.2 ld

the latency of ld:

Register Read (for PC) + I-Mem + RegFile + Mux + ALU + D-Mem + MUX

+ Register Setup (for RegFile)

$= (30 + 250 + 150 + 25 + 200 + 250 + 25 + 20)\,ps$

$= 950\,ps$

4.7.3 sd

the latency of sd

Register Read (for PC) + I-Mem + RegFile + Mux + ALU + D-Mem

$= (30 + 250 + 150 + 25 + 200 + 250)\,ps = 905\,ps$

4.7.4 beq

Register Read (for PC) + I-Mem + RegFile + Mux + ALU + (+single gate) Mux + Register Setup (for PC)

$= (30 + 250 + 150 + 25 + 200 + 5 + 25 + 20)\,ps = 705\,ps$

4.7.5 I-type { ALU-RI, load, jalr }

load: 950ps

ALURI:

Register Read (for PC) + I-Mem + RegFile + MUX + ALU + MUX + Register Setup (for RegFile)

$= (30 + 250 + 150 + 25 + 200 + 25 + 20)\,ps = 700\,ps$

jalr:

Register Read (for PC) + I-Mem + RegFile + Mux + ALU + Mux + Register Setup (for PC)

$= (30 + 250 + 150 + 25 + 200 + 25 + 20)\,ps = 700\,ps$

4.7.6 the longest latency is lw. 950 ps

∴ the minimum clock period is 950 ps

4.9

4.9.1 假设使用4.7中给定的参数

如4.7所示，在没有修改ALU之前，clock cycle time = 950 ps

修改之后，clock cycle time = (950+300)ps = 1250ps

4.9.2 修改前 时间 IC×950

修改后 时间 IC×95%×1250

$$\text{speedup} = \frac{IC \times 950}{IC \times 95\% \times 1250} = 0.8$$

4.9.3 $IC \times 95\% \times (950 + \Delta t) = IC \times 950$

$\Delta t = 50\,ps$

the slowest new ALU can be 350ps

4.13 SS rs1, rs2, imm Mem[rs1] = rs2 + imm

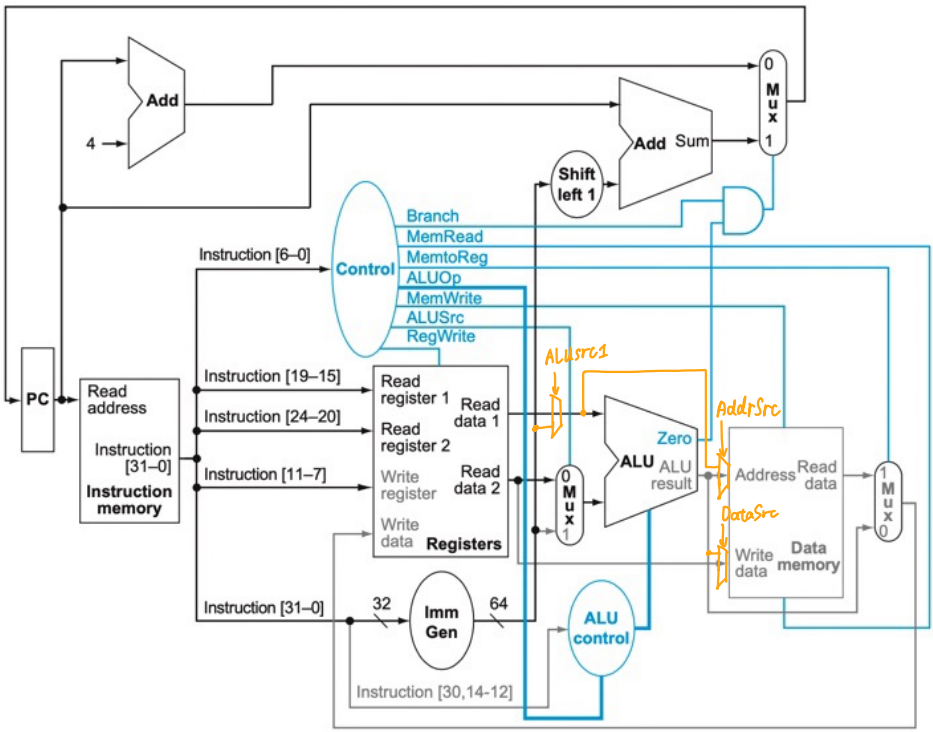


FIGURE 4.21 The datapath in operation for a branch-if-equal instruction. The control lines, datapath units, and connections that are active are highlighted. After using the register file and ALU to perform the compare, the Zero output is used to select the next program counter from between the two candidates.

4.13.1 No new functional block

4.13.2 No existing functional blocks to modify.

4.13.3 see the data path above

4.13.4 ALUsrc1 : 0 — rs1
1 — imm ✓

AddrSrc : 0 — ALU_res
1 — rs1 ✓

DataSrc : 0 — rs2
1 — ALU_res ✓

4.13.5 see the data path above

4.16

clock cycle time

4.16.1 pipelined processor : 350ps

un-pipelined processor : (250+350+150+300+200)ps = 1250ps

4.16.2 ld : (total latency)

pipelined = 1750ps

non-pipelined (also need all 5 stages) = 1250ps

4.16.3 split ID stage, because ID has the longest latency.

after splitting, there are 6 stages, with latency of 300ps each stage

new clock cycle time = 300ps

4.16.4 only load and store utilize DMem

∴ the utilization = 20% + 15% = 35%

4.16.5 ALU/Logic, Load utilize the write port of RegFile (actually, jalr also uses it)

∴ the utilization = 45% + 20% = 65% (不考虑jalr)

4.18

addi x11, x12, 5	IF	ID	EX	MEM	WB		
add x13, x11, x12		IF	ID	EX	MEM	WB	
addi x14, x11, 15			IF	ID	EX	MEM	WB

See the diagram. When executing add x13, x11, x12 and addi x14, x11, 15
x11 hasn't be modified yet.

x13 = 11 + 22 = 33

x14 = 11 + 15 = 26

4.20

```
addi x11, x12, 5
NOP
NOP
add x13, x11, x12
addi x14, x11, 15
NOP
add x15, x13, x12
```

4.23

4.23.1 If the latency of every stage is equal

Assume the cycle time of non-pipelined processor is $T_{unpipelined}$, after ALU and DMem parallel $T'_{unpipelined}$ ($< T_{unpipelined}$)

- when pipeline depth = 5, $T_{pipelined} \gtrsim \frac{T_{unpipelined}}{5}$
- when pipeline depth = 4, $T'_{pipelined} \gtrsim \frac{T_{unpipelined}}{4}$

the clock cycle time increases.

But if the latency of each stage is not uniform (like exercise 4.16)

then the clock cycle time doesn't change (the total latency decreases)

4.23.2 ① `lol x10, 0(x11)` IF ID EX WB

`addol x12, x10, x2` IF ID EX WB

can use forwarding path to solve the data hazard above, no need to stall

② reduce the latch cost

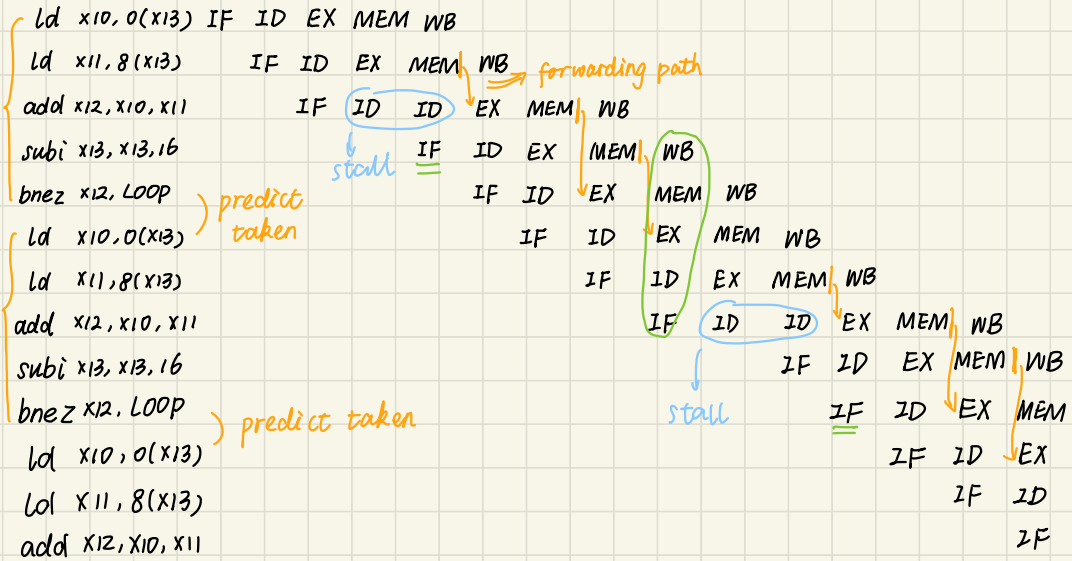
③ reduce the total latency of a single instruction

4.23.3 ① the clock cycle time might increase (due to the reduction of pipeline depth)
the throughput might decrease

② may add the total number of instructions
for the same operations

4.25

4.25.1



4.25.2 in the above diagram, I use ○ to mark the stages to stall.

ld x11, 8(x13) + add x12, x10, x11 must stall one stage.

I use ◇ to mark the time when the pipeline is full

we can observe that during each cycle, there is only one time when the pipeline is fully used.