CO 2021-Fall HW4 Solution

4.3.1

Load instruction and store instruction use data memory.
25% + 10% = 35%

4.3.2

All instructions use instruction memory.
24% + 28% + 25% + 10% + 11% + 2% = 100%

4.3.3

Except R-type instruction and jump instruction, all other instructions use the sign extend.
100% - 24% - 2% = 74%

4.3.4

The sign-extend circuit is actually computing a result in every cycle, but its output is ignored if its output is not needed.

4.5.1

ALU control unit's input is Instruction[5-0]: 100010

4.5.2

New PC = PC + 4

4.5.3

MUX	Input	Output
RegDst	In1: 00111 ($a3) in2: 10111 ($s7)	10111 ($s7)
ALUSrc	In1: Reg[$a3] in2: 0xFFFFB822	Reg[$a3]
MemtoReg	In1: undefined (read from DM) in2: Reg[$a2] - Reg[$a3]	Reg[$a2] - Reg[$a3]
Branch	In1: PC +4 in2: PC + 4 + 0xFFFEE088	PC +4
Jump	In1: {PC+4[31-28],0x31EE088} in2: PC +4	PC +4

4.5.4

ALU: Reg[$a2] and Reg[$a3]
Add1: PC and 4
Add2: PC+4 and 0xFFFEE088

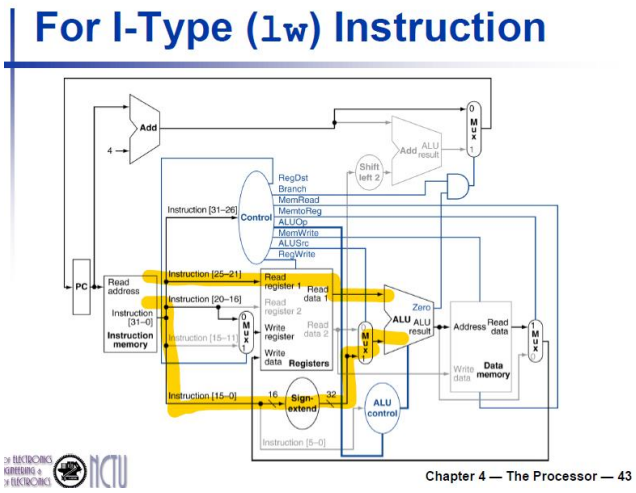
4.5.5

Read register1: 00110 ($a2)
Read register2: 00111 ($a3)
Write register: 10111 ($s7)
Write data: Reg[$a2] - Reg[$a3]

4.7.1

The latency of R-type instruction:
Register read + I-Mem + Register file + Mux + ALU + Mux + Register setup
30 + 250 + 150 + 25 + 200 + 25 + 20 = 700ps

4.7.2



The latency of lw instruction:
Register read + I-Mem + Register file + ALU + D-Mem + Mux + Register setup
(從 I-Mem 到 ALU 的路徑為 Register file 的 latency 最長)
30 + 250 + 150 + 200 + 250 + 25 + 20 = 925ps

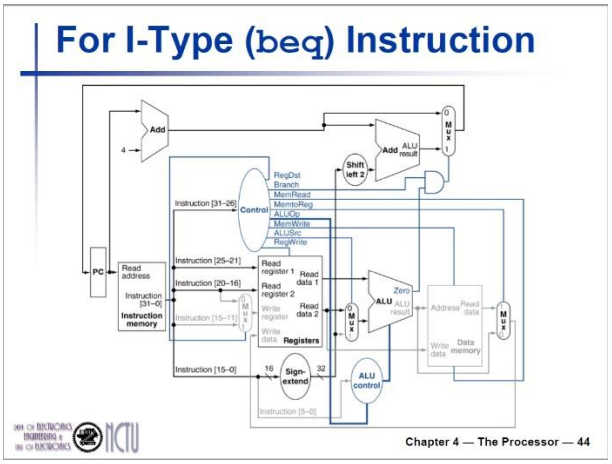
4.7.3

The latency of sw instruction:
Register read + I-Mem + Register file + ALU + D-Mem
30 + 250 + 150 + 200 + 250 = 880ps

4.7.4

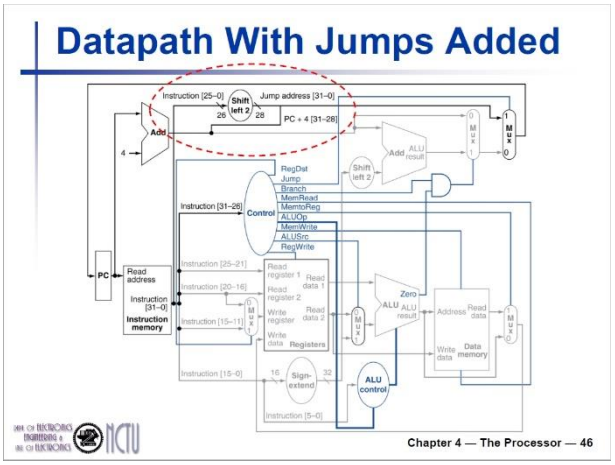
這題答案下面兩種都算對，到時考試時以完整的架構為準(有 jump 的)。

1. 沒支援 jump (依照講義給的 beq instruction)



The latency of beq instruction:
Register read + I-Mem + Register file + Mux + ALU + Single gate + Mux + Register setup
30 + 250 + 150 + 25 + 200 + 5 + 25 + 20 = 705ps

2. 有支援 jump



The latency of beq instruction:
Register read + I-Mem + Register file + Mux + ALU + Single gate + Mux + Mux + Register setup
30 + 250 + 150 + 25 + 200 + 5 + 25 + 25 + 20 = 730ps

4.7.5

The latency of I-type instruction:
Register read + I-Mem + Register file + ALU + Mux + Register setup
(從 I-Mem 到 ALU 共有三條路徑，分別為 control -> ALUSrc mux、Register file 以及 sign extend -> ALUSrc mux，其中 Register file 的 latency 最長，因此選 Register file)
30 + 250 + 150 + 200 + 25 + 20 = 675ps

4.7.6

Minimum clock period for this CPU is 925ps.

4.16.1

Pipelined: 350ps (longest execution time)
Non-pipelined: 250 + 350 + 150 + 300 + 200 = 1250ps

4.16.2

Pipelined: 350 * 5 = 1750ps
Non-pipelined: 250 + 350 + 150 + 300 + 200 = 1250ps

4.16.3

1. Stage to split: ID stage
2. New clock cycle time: 300ps

4.16.4

Load + Store = 20% + 15% = 35%

4.16.5

ALU/Logic + Load = 45% +20% = 65%

4.27.1

```
add   $s3, $s1, $s0
nop
nop
lw    $s2, 4($s3)
lw    $s1, 0($s4)
nop
or    $s2, $s3, $s2
nop
nop
sw    $s2, 0($s3)
```

4.27.2

```
add   $s3, $s1, $s0
lw    $s1, 0($s4)
nop
lw    $s2, 4($s3)
nop
```

```
nop
or    $s2, $s3, $s2
nop
nop
sw    $s2, 0($s3)
```

→ there is no performance gain

4.27.3

With forwarding, the hazard detection unit is still needed because it must insert a one-cycle stall whenever the load supplies a value to the instruction that immediately follows that load. Without the hazard detection unit, the instruction that depends on the immediately preceding load gets the stale value the register had before the load instruction.

4.27.4

The outputs of the hazard detection unit are PCWrite, IF/IDWrite, and ID/EXZero (which controls the Mux after the output of the Control unit). Note that IF/IDWrite is always equal to PCWrite, and ED/ExZero is always the opposite of PCWrite. As a result, we will only show the value of PCWrite for each cycle. The outputs of the forwarding unit is ALUin1 and ALUin2, which control Muxes that select the first and second input of the ALU. The three possible values for ALUin1 or ALUin2 are 0 (no forwarding), 1 (forward data value for second-previous instruction), or 2 (forward ALU output from previous instruction).

Instruction Sequence	First Seven Cycle						
	1	2	3	4	5	6	7
`add $s3, $s1, $s0`	IF	ID	EX	MEM	WB		
`lw $s2, 4($s3)`		IF	ID	EX	MEM	WB	
`lw $s1, 0($s4)`			IF	ID	EX	MEM	WB
`or $s2, $s3, $s2`				IF	ID	EX	MEM
`sw $s2, 0($s3)`					IF	ID	EX

Signals\Cycles	First Seven Cycle						
	1	2	3	4	5	6	7
PCWrite	1	1	1	1	1	1	1
ALUin1 (Rs)	X	X	0	2	0	0	0
ALUin2 (Rt)	X	X	0	0	0	1	2

4.27.5

The instruction that is currently in the ID stage needs to be stalled if it depends on a value produced by the instruction in the EX or the instruction in the MEM stage. So we need to check the destination register of these two instructions. For the instruction in the EX stage, we need to check Rd for R-type instructions and Rd for loads. For the instruction in the MEM stage, the destination register is already selected (by the Mux in the EX stage) so we need to check that register number (this is the bottommost output of the EX/MEM pipeline register). The additional inputs to the hazard detection unit are register Rd from the ID/EX pipeline register and the output number of the output register from the EX/MEM pipeline register. The Rt field from the ID/EX register is already an input of the hazard detection unit in Figure 4.60. No additional outputs are needed.

We can stall the pipeline using the three output signals that we already have.

4.28.1

$$CPI_{extra} = 1 * 0.55 * 0.25 = 0.1375$$

4.28.2

$$CPI_{extra} = 1 * 0.45 * 0.25 = 0.1125$$

4.28.3

$$CPI_{extra} = 1 * (1 - 0.85) * 0.25 = 0.0375$$

4.28.4

$$CPI_{without\ conversion} = 1 + 1 * (1 - 0.85) * 0.25 = 1.0375$$

$$CPI_{with\ conversion} = 1 + 1 * (1 - 0.85) * (0.25 * 0.5) = 1.01875$$

$$Speed\ up = \frac{1.0375}{1.01875} = 1.0184$$

4.28.5

$$CPI_{without\ conversion} = 1 + 1 * (1 - 0.85) * 0.25 = 1.0375$$

$$CPI_{with\ conversion} = 1 + 1 * (1 - 0.85) * (0.25 * 0.5) + (0.25 * 0.5) = 1.14375$$

$$Speed\ up = \frac{1.0375}{1.14375} = 0.907$$

4.28.6

$$Correctly\ Predicted_{total} = 0.85$$

$$Correctly\ Predicted_{non-look-back} = 0.85 - 0.8 = 0.05$$

$$Accuracy_{non-look-back} = \frac{0.05}{1 - 0.8} = 0.25$$

4.29.1

$$Accuracy_{taken} = \frac{3}{5} = 0.6$$

$$Accuracy_{non-taken} = \frac{2}{5} = 0.6$$

4.29.2

Outcomes	Prediction	Correct or Incorrect
T	NT (0)	InCorrect
NT	NT (1)	Correct
T	NT (0)	InCorrect
T	NT (1)	InCorrect

$$Accuracy = \frac{1}{4} = 0.25$$

4.29.3

Outcomes	Prediction for steady state	Correct or Incorrect
T	T (2)	Correct
NT	T (3)	InCorrect
T	T (2)	Correct
T	T (3)	Correct
NT	T (3)	InCorrect

$$Accuracy = \frac{3}{5} = 0.6$$

4.29.4

The predictor should be an N-bit shift register, where N is the number of branch outcomes in the target pattern. The shift register should be initialized with the pattern itself (0 for NT, 1 for T), and the prediction is always the value in the left most bit of the shift register. The register should be shifted after each predicted branch.

4.29.5

Since the predictor's output is always the opposite of the actual outcome of the branch instruction, the accuracy is zero.

4.29.6

The predictor is the same as in 4.29.4, except that it should compare its prediction to the actual outcome and invert (logical NOT) all the bits in the shift register if the prediction is incorrect. This predictor still always perfectly predicts the given pattern. For the opposite pattern, the first prediction will be incorrect, so the predictor's state is inverted and after that the predictions are always correct. Overall, there is no warm-up period for the given pattern, and the warm-up period for the opposite pattern is only one branch.