CS 331 Final Exam Review

__Overview__

Chapter 4

- Pipelined Processor
  - The basic concepts of pipelining
  - Design of pipelined datapath and control
  - Data flow of various MIPS instructions and their associated control signals
  - Detect data hazards, and handle them with/without data forwarding
    - How many pipeline stalls are needed with / without forwarding?
  - Detect and handle control hazard with prediction and pipeline flush
    - Static, 1 - bit and 2 – bit predictor
  - Performance and speedup of pipelined datapath
    - Cycle time, CPI, split into more stages?

See: outline from Alexis

Chapter 5

- Basic concepts of memory Hierarchy
  - Principle of Locality (Temporal and Spatial)
- Basics of Cache Design
  - Direct mapped cache, set associative, fully associative
  - One word vs. Multi-word blocks
  - What happens with byte address, word address, and block address
  - How to compute cache index, tag, block offset, byte offset
  - Write through vs. Write back
  - Cache Replacement (LRU)
  - Multi-level caches
- Measuring and analyzing cache performance
  - Stall cycles, CPI based on miss rate, miss penalty, AMAT
  - For different designs, with or without secondary level cache
  - Ways to reduce 3Cs

See: outline from Alexis

Practice Exercises for Exam 3

Question 1

Consider the 5-stage pipelined MIPS datapath that implements a subset of the MIPS instruction set (R-format, Lw, Sw, beq). Assume the following latencies for the major functional units: 200 ps for instruction memory access, 150 ps for ALU operation, and 100 ps for register file access and 300 ps for data memory access.

1. What is the shortest clock cycle?
2. What is the latency of a sw instruction?
3. If we split one of the 5-stages into 2 equal stages in the pipelined design to decrease the clock cycle, which stage should we split? What would be the new clock cycle?

Given: instruction memory = 200 ps, ALU = 150 ps, registers = 100 ps, data memory = 300 ps The shortest clock cycle is based on the longest stage. Therefore, the shortest clock cycle is 300 ps.
In a **pipelined** datapath, the latency from data memory access affects all 5-stages. So latency is: 300 ps + 300 ps + 300 ps + 300 ps + 300 ps = 1500 ps. **Note**: If this **wasn't pipelined**, consider the major functional units required to execute the **lw instruction** (the critical path): 150 + 100*2 + 300 + (200) = 850 ps
The longest stage is split, so 300/2 ps = 150 ps. The new clock cycle would be the longest stage. Therefore, the shortest clock cycle is now 200 ps.

Question 2

Consider the following MIPS segment code when answering the questions below:

```
sw  $t5, 0($t2)
sub $t3, $t1, $t6
```

```
lw  $t1, 12($t5)
or  $t5, $t1, $t3
```

1. Consider running this code on the 5-stage pipelined processor that does not support forwarding and hazard detection. What is the total number of cycles required to execute this code?
2. Consider running this code on the 5-stage pipelined processor that supports forwarding and hazard detection.
   a. What is the total number of cycles required to execute this code?
   b. How many stalls?
   c. How many forwards?
   d. What is ALU doing during cycle 7?
   e. Can we rearrange the code to minimize the number of cycles and speed up the execution?

```
sw  $t5, 0($t2)    (memory[t2+0] = t5)
sub $t3, $t1, $t6  (t3 = t1 + t6)
lw  $t1, 12($t5)   (t1 = memory[t5+12]; RAR not a RAW)
or  $t5, $t1, $t3  (t5 = t1 | t3; stall until correct $t1 is read)

01 02 03 04 05 06 07 08 09 10
IF ID EX MM WB                (updates t5 in mem stage)
-- IF ID EX MM WB             (no dependencies, so no stalls)
-- -- IF ID EX MM WB          (updates t1 in wb stage)
-- -- -- -- -- IF ID EX MM WB (depends on t1, so stall twice to ex after wb)
```

Note: sw updates register in mem stage, lw and R-type instructions update register in write-back.

With 2 stalls, a total of 10 cycles is required to execute this code.

Since the lw instruction stores memory[t5+12] into register t1, the value memory[t5+12] is obtained in the **memory stage**. The execution stage only calculates the address (t5+12).

As a result, one stall is needed before forwarding the value from the **memory stage** to the dependent instruction's **execution stage**.

```
01 02 03 04 05 06 07 08 09
IF ID EX MM WB
-- IF ID EX MM WB
-- -- IF ID EX MM WB       (lw forwards value t1 from memory stage)
-- -- -- -- IF ID EX MM WB (depends on t1)
```

With 1 forward and 1 stall, a total of 9 cycles is required to execute this code.

In cycle 7, the or instruction is in the execute (EX) stage. Therefore, the operands in registers t1 and t3 are evaluated in the or functionality of the ALU.

Reorder to save one stall cycle: Sub, lw, sw, or

```
sub $t3, $t1, $t6  (t3 = t1 + t6)
lw  $t1, 12($t5)   (t1 = memory[t5+12])
sw  $t5, 0($t2)    (memory[t2+0] = t5)
or  $t5, $t1, $t3  (t5 = t1 | t3)
```

Different example from office hour:

```
sw  $t5, 0($t2)
sub $t3, $t1, $t6
lw  $t4, 12($t5)   (RAR not a RAW)
or  $t5, $t1, $t3  (stall until you read the correct $t1)

IF ID EX MM WB                (updates t5 in mem stage)
-- IF ID EX MM WB             (no dependencies, so no stalls)
-- -- IF ID EX MM WB          (updates t1 in wb stage)
-- -- -- -- IF ID EX MM WB    (depends on t3, so one stall is required)
```

Question 3

Assume branches constitute 50% of the instructions of a program running on a 5-stage MIPS pipeline, branch outcome is determined in the Execute stage and there are no data hazards when answering the questions below.

1. How many cycles will be lost on a mispredicted branch?
2. What is the CPI or extra CPI due to mispredicted branches if the predictor has a 25% branch prediction accuracy?

If the branch outcome is determined to be false in the execute (EX) stage, then two cycles are lost (IF, ID).

When the branch outcomes are determined at the EX stage, The branch CPI = 1 if prediction is correct and branch CPI = 3 if the prediction is wrong.

Average branch CPI for always-taken prediction (25% accuracy) = 1×25% + 3×75% = 2.5 cycles/instruction.

```
Average CPI = (constitutes% x baseCPI) + (constitutes% x avg branch CPI)
Average CPI = 50%x1 + 50%x2.5 = 1.75 cycles/instruction.

Extra CPI = Average CPI - Base CPI
Extra CPI = 1.75 - 1 = 0.75 cycles/instruction.

Another way to see CPI:
```

Extra CPI = lost cycles x %mispredicted x Frequency of Branch instructions
Extra CPI = 2x75% x 50% = 0.75 cycles per instruction

```
IF ID EX M WB
   IF ID
      IF
      IF ID EX M WB
```

Question 4

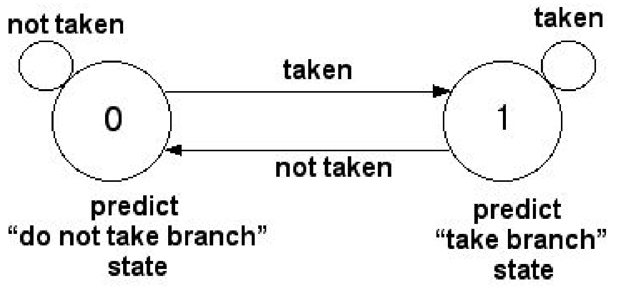
For the following 3 sequences of branch execution patterns, assume initial predict taken for 1-bit predictor and strongly taken for 2-bit predictor. What is the accuracy of each predictor (percentage)?

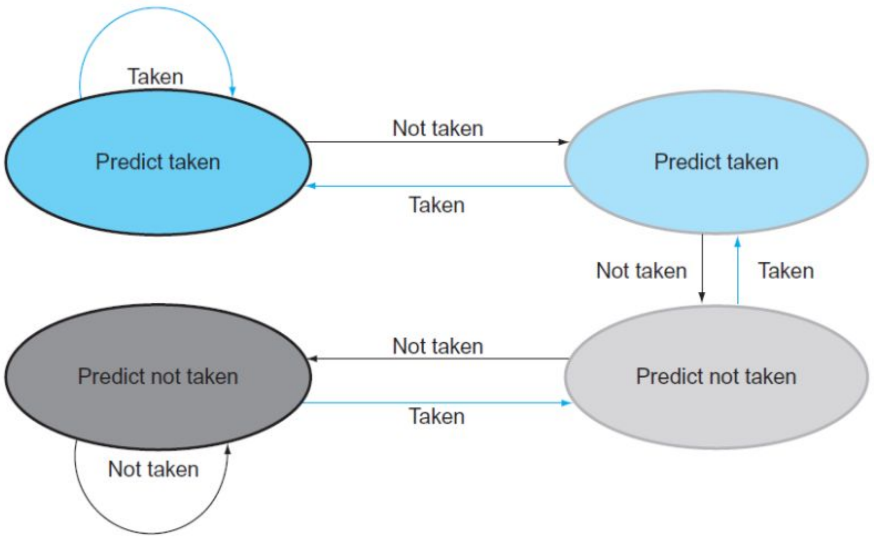
If you are doing a bit predict for every NT you will have 2 misses

0 ============= 1

	1-bit	2-bit
T T T T NT	4/5 = 80%	4/5 = 80%
T T NT T NT	2/5 = 40%	3/5 = 60%
NT T NT T NT	0/5 = 0%	2/5 = 40%

1-bit:





2-bit:

Given Outcomes	1-bit Prediction	2-bit Prediction

T T T T NT	T = in “1”, stay in “1” (h)
	T = in “1”, stay in “1” (h)
	T = in “1”, stay in “1” (h)
	T = in “1”, stay in “1” (h)
	NT = in “1”, go to “0” (m)
		T = in strongly taken, stay in strongly taken (h)
		T = in strongly taken, stay in strongly taken (h)
		T = in strongly taken, stay in strongly taken (h)
		T = in strongly taken, stay in strongly taken (h)
		NT = in strongly taken, go to weakly taken (m)

T T NT T NT	T = in “1”, stay in “1” (h)
	T = in “1”, stay in “1” (h)
	NT = in “1”, go to “0” (m)
	T = in “0”, go to “1” (m)
	NT = in “1”, go to “0” (m)
		T = in strongly taken, stay in strongly taken (h)
		T = in strongly taken, stay in strongly taken (h)
		NT = in strongly taken, go to weakly taken (m)
		T = in weakly taken, go to strongly taken (h)
		NT = in strongly taken, go to weakly taken (m)

NT T NT T NT	NT = in “1”, go to “0” (m)
	T = in “0”, go to “1” (m)
	NT = in “1”, go to “0” (m)
	T = in “0”, go to “1” (m)
	NT = in “1”, go to “0” (m)
		NT = in strongly taken, go to weakly taken (m)
		T = in weakly taken, go to strongly taken (h)
		NT = in strongly taken, go to weakly taken (m)
		T = in weakly taken, go to strongly taken (h)
		NT = in strongly taken, go to weakly taken (m)

Question 5

Consider a sequence of memory references of word addresses:

22, 25, 24, 56, 40, 24, 20
Cache size: 8 words
LRU replacement strategy

Consider 3 different cache organizations:

Direct mapped, block size: 1 word

2-way set associative, block size: 2 words
Fully associative, block size: 1 word

For each organization:

1. Specify the number of index/offset bits.
2. Mark each reference as hit (h) or miss (m) and show the final content of the cache.
3. When done, discuss different techniques that help reduce the miss rate.

Solution: see Module 6 Assessment

Question 6

Suppose we have a processor with a base CPI of 1.0, assuming all references hit in the primary cache, and a clock rate of 4 GHz. Assume a main memory access time of 100 ns. Suppose the miss rate at the primary cache is 2%.

1. How much faster will the processor be if we add a secondary cache that has a 5 ns access time and is large enough to reduce the miss rate to main memory to 0.5% (compare CPIs)?
2. What is the AMAT when having L1 only or L1-L2 caches?

Given:
Base CPI = 1.0
CR = 4 GHz)
CT = 1/(4 GHz) = 0.25 ns (note: nano = 10E-9 units, Giga = 10E+9 units)
L1 miss rate = 0.02
Main memory access time (CPU->L1) = 100 ns
L2 access time = 5 ns
L2 miss rate = 0.005

Use L1:
CPI = Base CPI + MCPI
Memory CPI, MCPI = MR * MP
Miss penalty, MP = memory access time / CT
CT = inverse of CR

Use L2:
CPI = Base CPI + (MR[L1] * MP[L2]) + (MR[L2] * MP[memory])

<u>L1 Cache</u>
CPI = Base CPI + Memory CPI

CPI = 1.0 + MCPI

MCPI = Miss Rate(MR) * Miss Penalty(MP)

MP = memory access time / CT
MP = memory access time / (1/CR)
MP = 100 ns / (0.25 ns) = 400 cycles

Memory CPI = (0.02) * (400 cycles) = 8 cycles/instruction

CPI = 1.0 + 8 = 9 cycles/instruction

<u>L2 Cache</u>

CPI = (1.0) + (0.02) * MP(L2) + (0.005) * MP(memory)

MP = memory access time / CT
MP(L2) = 5 ns / (0.25 ns) = 20 cycles

MP(memory) = 400 cycles

CPI = (1.0) + ((0.02) * (20 cycles)) + ((0.005) * (400 cycles))
CPI = 3.4 cycles/instruction

Improvement = (CPI_L1 / CPI_L2) = (9 / 3.4) = 2.6 times faster

---

Use:
1-Level Cache:
AMAT = Hit Time + (MR * MP)
2-Level Cache:
AMAT = Hit Time(L1) + (MR(L1) * MP(L2)) + (MR(L2) * MP(Main Memory))

2 ns == 1 cycle
Frequency = CR = 4 GHz so CT = 1/frequency = 1/4GHz = 0.25 ns

<u>L1 Cache</u>
AMAT = Hit Time + MR * MP
AMAT = 0.25 + (0.02 * 400 cycles)
AMAT = 0.25 + (8)
AMAT = 8.25
Hit time is either 1 cycle or 1/4GHz = 0.25 ns
If you use hit time in ns (0.25 ns) then you use MP in ns.
If you use hit time as 1 cycle then you use MP(L2) as 5ns/0.25 and MP (memory) as 100ns/0.25

<u>L2 Cache</u>
AMAT = Hit Time + (MR(L1) * MP(L2)) + (MR(L2) * MP(Main Memory))
AMAT = 0.25 + (0.02 * 5 ns) + (0.005 * 400 cycles) = 2.35

Module 5, Clock Cycle Assessment

Consider the basic single-cycle MIPS datapath that implements a subset of the MIPS instruction set (R-format, Lw, Sw, beq). Assume the following latencies for the major functional units: 220 ps for memory access, 70 ps for ALU operation, 60 ps for register read, and 100 ps for register write.

Question 1

What is the shortest clock cycle for this single cycle datapath?

670 ps
570 ps
450 ps
730 ps

For a **single cycle** datapath, consider **the most expensive operation**- the lw instruction- as the critical path.

Clock cycle time = (memory access) + (register read) + (ALU offset math) + (memory access) + (register write)

Clock cycle time = 220 + 60 + 70 + 220 + 100 = 670 ps.

Question 2

What is the shortest clock cycle if we implement a 5 stage pipelined design?

220 ps
670 ps
100 ps
320 ps

For a **5-stage pipelined** design, consider **the longest stage only**.

Since memory access is the slowest stage, its latency of 220 ps is the shortest possible clock cycle time.

Module 5, Forwarding Assessment

Consider the following MIPS segment code:

```
    sub $2, $1, $3
    and $4, $2, $5
    or $4, $4, $2
    add $9, $4, $2
```

Question 1

Consider running this code on the 5-stage pipelined processor that does not support forwarding. What is the total number of cycles required to execute this code?

14

4

16

8

```
sub $2, $1, $3 (r2 = r1 - r3)
and $4, $2, $5 (r4 = r2 & r5)
or $4, $4, $2  (r4 = r4 | r2)
add $9, $4, $2 (r9 = r4 + r2)

01 02 03 04 05 06 07 08 09 10 11 12 13 14
IF ID EX MM WB
-- -- -- IF ID EX MM WB  (dependency on r4; ex after prev. wb = 2 stalls)
-- -- -- -- -- -- IF ID EX MM WB  (dependency on r4)
-- -- -- -- -- -- -- -- -- IF ID EX MM WB  (dependency on r4)

Total number of cycles = 14.
```

In this case: without forwarding, the dependent instruction's execution (EX) stage occurs after the previous instruction's write back (WB) stage.

Question 2

Consider running this code on the 5-stage pipelined processor that supports forwarding. What is the total number of cycles required to execute this code?

Select one:

8

11

16

14

```
sub $2, $1, $3 (r2 = r1 - r3)
and $4, $2, $5 (r4 = r2 & r5)
or $4, $4, $2  (r4 = r4 | r2)
add $9, $4, $2 (r9 = r4 + r2)

IF ID EX MM WB
-- IF ID EX MM WB  (dependency on r4; ex after prev. ex)
-- -- IF ID EX MM WB  (same dependency on r4)
-- -- -- IF ID EX MM WB  (same dependency on r4)

Total number of cycles = 8.
```

In this case: with forwarding, the dependent instruction’s execution (EX) stage occurs after the previous instruction’s execution (EX) stage.

All the above operations (sub, and, or, add) produce the requested data in the ALU, so said data can be forwarded from the execution (EX) stage.

Module 5, Hazard Detection Unit Assessment

Consider the following MIPS segment code when answering the questions below:

```
lw $2, 20($1)
and $4, $2, $5
or $4, $4, $2
add $9, $4, $2
```

Question 1

Consider running this code on the 5-stage pipelined processor that does not support forwarding and hazard detection. What is the total number of cycles required to execute this code?

14 cycles
11 cycles
12 cycles
9 cycles

```
lw $2, 20($1)  (r2 = memory[r1+20])
and $4, $2, $5 (r4 = r2 & r5)
or $4, $4, $2  (r4 = r4 | r2)
add $9, $4, $2 (r9 = r4 + r2)
```

```
01 02 03 04 05 06 07 08 09 10 11 12 13 14
IF ID EX MM WB
-- -- -- IF ID EX MM WB  (dependency on r2; ex after prev. wb = 2 stalls)
-- -- -- -- -- -- IF ID EX MM WB  (dependency on r4)
-- -- -- -- -- -- -- -- -- IF ID EX MM WB (dependency on r4)
```

Total number of cycles = 14.

In this case: without forwarding, the dependent instruction's execution (EX) stage occurs after the previous instruction's write back (WB) stage.

Question 2

Consider running this code on the 5-stage pipelined processor that supports forwarding and hazard detection. What is the total number of cycles required to execute this code?

9

11

8

14

```
lw $2, 20($1)  (r2 = memory[r1+20])
and $4, $2, $5 (r4 = r2 & r5)
or $4, $4, $2  (r4 = r4 | r2)
add $9, $4, $2 (r9 = r4 + r2)

01 02 03 04 05 06 07 08 09
IF ID EX MM WB
-- IF ID EX MM WB  (dependency on r2; ex after prev. ex)
-- -- -- IF ID EX MM WB  (dependency on r4; lw = ex after prev. mem = 1 stall)
-- -- -- -- IF ID EX MM WB (dependency on r4)
```

Total number of cycles = 9.

In this case: the lw instruction determines the address (r1+20) through the ALU in the execution (EX) stage. The actual value from memory[r1+20] is obtained in the memory (MM) stage.

As a result, the next dependent instruction will stall once to allow for its execution stage (EX) to receive the lw instruction's value forwarded from its memory (MM) stage.

Question 3

Consider running this code on the 5-stage pipelined processor that supports forwarding and hazard detection. How many forwards are needed to ensure the correct execution of this code sequence?
3
0
1
2

Based on the previous solution:

```
lw $2, 20($1)  (r2 = memory[r1+20])
and $4, $2, $5 (r4 = r2 & r5)
or $4, $4, $2  (r4 = r4 | r2)
add $9, $4, $2 (r9 = r4 + r2)

01 02 03 04 05 06 07 08 09
IF ID EX MM WB
-- IF ID EX MM WB  (dependency on r2; ex after prev. ex)
-- -- -- IF ID EX MM WB  (dependency on r4; lw = ex after prev. mem = 1 stall)
-- -- -- -- IF ID EX MM WB (dependency on r4)
```

There were 3 forwards (r2 from *lw* to *and*, r4 from *and* to *or*, r4 from *or* to *add*).

Module 6, Basic DM Cache Assessment

Consider a series of address references given as word addresses: 22, 24, 25, 20 Assuming a 16-word direct-mapped cache one-word blocks, compute the cache index of each reference and label it as hit or miss. then answer the following questions.

Question 1 to Question 4

Find the cache index for the following words: 22, 24, 25, 20.

```
Direct Mapped, block size = 1 word

In decimal...
Cache index = (block address) % (number of cache blocks)

22: cache index = 22 % 16 = 6
24: cache index = 24 % 16 = 8
25: cache index = 25 % 16 = 9
20: cache index = 20 % 16 = 4

6, 8, 9, 4.
```

Module 6, Cache Performance Assessment

Question 1

Assume the miss rate of an instruction cache is 2%, the miss rate of the data cache is 4% and the frequency of all loads and stores is 36%. The processor has a base CPI of 2 without any memory stalls and the miss penalty is 100 cycles for all misses.

If we double the clock rate without changing the memory system, what would be the new overall CPI? And how much is the speedup?

CPI > 5.44 and speedup < 2

CPI > 5.44 and speedup > 2

CPI < 5.44 and speedup < 2

CPI < 5.44 and speedup > 2

```
Faster clock cycles means Miss penalty twice as long: Miss penalty = 200 cycles
Total miss cycle: total number of memory stalls: 2% × 200 × IC + 36% × 4% × 200
× IC = 6.88 IC
Actual CPI with memory stalls= 2 + 6.88 = 8.88

So Speedup = 5.44 × 2 / 8.88 = 1.23
the performance loss due to cache misses increased
The correct answers are: CPI > 5.44 and speedup < 2, CPI > 5.44 and speedup > 2,
CPI < 5.44 and speedup < 2, CPI < 5.44 and speedup > 2
```

Module 6, Alternative Cache Designs Assessment

Solution: see Module 6 Assessment

Module 5, Branch Prediction Assessment

Question 1

Assume the following execution pattern for a given branch: T NT T NT T NT.

Assume branches constitute **28%** of the instructions, branch outcome is determined in the Memory stage and there are no data hazards. What is the overall CPI when using the always taken predictor?

1.42
2.5
2
1.28
None of these answers

prediction: T T T T T T

so prediction accuracy = **50%**

branch outcome determined in Memory stage so 0 lost cycles per branch instruction if right, 3 if wrong

we are right 50% of time so CPI of branch = (1 × 0.5 + 4 × 0.5) = 2.5
CPI of Branch =
Base CPI + prediction accuracy + (missed stages + 1) x prediction accuracy

If 28% of instructions are Branches and CPI of other instructions=1
Total CPI = 2.5 × 0.28 + 1 × 0.72 = 1.42
Total CPI = Branch CPI x branchConst + baseCPI x (100 - branchConst)

Extra CPI = lost cycles x %mispredicted x Frequency of Branch instructions

---

Poll Everywhere Questions (11/02 - 11/30)

Question 11.02.1

How many cycles does it take to execute this sequence?

```
lw  $10, 20 ($1)
sub $11, $2, $3
and $12, $4, $5
or  $13, $6, $7
add $14, $8, $9
```

5
9
10
25

```
01 02 03 04 05 06 07 08 09
IF ID EX MM WB
-- IF ID EX MM WB
-- -- IF ID EX MM WB
-- -- -- IF ID EX MM WB
-- -- -- -- IF ID EX MM WB

Total number of cycles = 9.
```

Question 11.09.1

Consider this code on the 5-stage pipeline processor. How many cycles are needed without forwarding?

```
sub $2, $1, $3
and $4, $2, $5
or  $4, $4, $2
add $9, $4, $2
```

8
14
11
16

```
sub r2, r1, r3 (r2 = r1 - r3)
and r4, r2, r5 (r4 = r2 & r5)
or  r4, r4, r2 (r4 = r4 | r2)
add r9, r4, r2 (r9 = r4 + r2)

01 02 03 04 05 06 07 08 09 10 11 12 13 14
IF ID EX MM WB
-- -- -- IF ID EX MM WB  (dependency on r2; ex after prev. wb = 2 stalls)
-- -- -- -- -- -- IF ID EX MM WB  (dependency on r4)
-- -- -- -- -- -- -- -- -- IF ID EX MM WB  (dependency on r4)

Total number of cycles = 14.
```

Question 11.09.2

Consider this code on the 5-stage pipeline processor. How many cycles are needed with forwarding?

```
sub $2, $1, $3
and $4, $2, $5
or  $4, $4, $2
add $9, $4, $2
```

8
14
11
16

```
sub r2, r1, r3 (r2 = r1 - r3)
and r4, r2, r5 (r4 = r2 & r5)
or  r4, r4, r2 (r4 = r4 | r2)
add r9, r4, r2 (r9 = r4 + r2)

01 02 03 04 05 06 07 08
IF ID EX MM WB
-- IF ID EX MM WB  (dependency on r2; ex after prev. ex)
-- -- IF ID EX MM WB  (dependency on r4)
-- -- -- IF ID EX MM WB  (dependency on r4)

Total number of cycles = 8.
```

Question 11.13.1

How many cycles required assuming forwarding and hazard detection supported?

```
lw $2, 20($1)
and $4, $2, $5
or $4, $4, $2
add $9, $4, $2
```

8

[✓ 9]

11

14

```
lw  r2, 20(r1)  (r2 = memory[r1+20])
and r4, r2, r5  (r4 = r2 & r5)
or  r4, r4, r2  (r4 = r4 | r2)
add r9, r4, r2  (r9 = r4 + r2)

01 02 03 04 05 06 07 08 09
IF ID EX MM WB
-- -- IF ID EX MM WB  (dependency on r2; lw = ex after prev. mem = 1 stall)
-- -- -- IF ID EX MM WB  (dependency on r4; ex after prev. ex)
-- -- -- -- IF ID EX MM WB  (dependency on r4)
```

Total number of cycles = 9.

In this case: the lw instruction determines the address (r1+20) through the ALU in the execution (EX) stage. The actual value from memory[r1+20] is obtained in the memory (MM) stage.

As a result, the next dependent instruction will stall once to allow for its execution stage (EX) to receive the lw instruction’s value forwarded from its memory (MM) stage.

Question 11.13.2

How many values require hardware forwarding?

```
lw $2, 20($1)
and $4, $2, $5
or $4, $4, $2
add $9, $4, $2
```

1 forward

2 forwards

[✓ 3 forwards]

no forward

Based on the previous solution:

Question 11.13.3

How many stalls occur? (forwarding/hazard detection supported)

```
add $3, $2, $1
lw  $4, 100($3)
and $6, $4, $3
sub $7, $6, $2
add $9, $3, $6
```

[✓ 1 stall]
2 stalls
3 stalls
No stalls

Question 11.13.4

How many values require hardware forwarding support to avoid stalling for our MIPS 5-stage pipeline?

```
add $3, $2, $1
lw  $4, 100($3)
and $6, $4, $3
sub $7, $6, $2
add $9, $3, $6
```

1 forward
2 forwards
3 forwards
[✓ 4 forwards]

Question 11.16.1

How many cycles are required? (forwarding/hazard detection supported)

```
sw   $t2, 20($t0)
add  $t3, $t1, $t4
lw   $t1, 4($t2)
and  $t2, $t1, $t3
```

8 cycles
9 cycles [✓]
14 cycles
15 cycles

Question 11.16.2

What is ALU doing during cycle 4?

```
sw   $t2, 20($t0)
add  $t3, $t1, $t4
lw   $t1, 4($t2)
and  $t2, $t1, $t3
```

Computing sw address
Computing lw address
Anding $t1 and $t3
Adding $t1 and $t4 [✓]

Question 11.30.1

Acceleration of CPU: If we reduce the ideal CPI from 2 to 1 without changing the clock rate or memory system, CPI = ?, speedup = ?
CPI < 5.44, Speedup > 2 [✓]
CPI > 5.44, Speedup < 2
CPI < 5.44, Speedup = 2

CPI > 5.44, Speedup = 2
None of the above