NOTE: this is the SOLUTION to Exam 1.

The correct answers are indicated for each question, with explanations as needed.

Dr. Manikas

1 4 / 4 points

Classes of architectural parallelism include:

- [ ] Task-Level Parallelism (TLP)
- [ ] Data-Level Parallelism (DLP)
- [x] Thread-Level Parallelism
- [x] Instruction-Level Parallelism (ILP)

Feedback

General Feedback

- Classes of parallelism in applications:
  - Data-Level Parallelism (DLP)
  - Task-Level Parallelism (TLP)
- **Classes of architectural parallelism:**
  - Instruction-Level Parallelism (ILP)

– Thread-Level Parallelism

2 4 / 4 points

Classes of parallelism in applications include:

- [x] Data-Level Parallelism (DLP)
- [ ] Thread-Level Parallelism
- [x] Task-Level Parallelism (TLP)
- [ ] Instruction-Level Parallelism (ILP)

Feedback

General Feedback

- **Classes of parallelism in applications:**
  - Data-Level Parallelism (DLP)
  - Task-Level Parallelism (TLP)
- Classes of architectural parallelism:
  - Instruction-Level Parallelism (ILP)
  - Thread-Level Parallelism

3 4 / 4 points

Your design team wishes to manufacture a new microprocessor, with a codename of "Peruna". The chip has a die size of 400 mm^2, with an estimated defect rate of 0.02/cm^2. Your manufacturing facility has process-complexity factor of 10, and we are assuming a wafer yield of 100%. *What is the **yield** for the Pernuna chip?*

- [x] 0.463

Feedback

General Feedback

Recall that we can calculate die yield using the following equation:

$$DieYield = \frac{WaferYield}{(1+(DefectsPerUnitArea)(DieArea))^N}$$

Then we have:

Wafer yield = 100% = 1

Defects per unit area = 0.02/cm^2

Die area = 400 mm^2

Process-complexity factor N = 10

Need to convert area so that units match: 1 cm = 10 mm, so (1 cm)^2 = (10 mm)^2, or 1 cm^2 = 100 mm^2

$$DefectsPerUnitArea = \frac{0.02}{cm^2}[\frac{cm^2}{100mm^2}] = \frac{0.02}{100mm^2}$$

Now we can plug everything into the equation:

$$DieYield = \frac{1}{(1+(\frac{0.02}{100})(400))^{10}} = \frac{1}{(1+0.08)^{10}} = 0.463$$

4 4 / 4 points

Your design team wishes to manufacture a new microprocessor, with a codename of "Pony". The chip has a die size of 200 mm^2. Your manufacturing facility makes wafers that are 400 mm in diameter. *How many Pony dies can you get from one wafer?*

✓ 565

Feedback

General Feedback

$$DiesPerWafer = \frac{\pi[\frac{WaferDiameter}{2}]^2}{DieArea} - \frac{\pi(WaferDiameter)}{\sqrt{2(DieArea)}}$$

Plugging in our values, we get:

$$DiesPerWafer = \frac{\pi[\frac{400}{2}]^2}{200} - \frac{\pi(400)}{\sqrt{2(200)}}$$

$$DiesPerWafer = \pi[\frac{200^2}{200} - \frac{400}{\sqrt{400}}] = 180\pi$$

This rounds *down* to **565** dies per wafer, since we are interested in producing **complete** dies.

5 4 / 4 points

Your design team wishes to manufacture a new microprocessor, with a codename of "Goose". Your manufacturing facility can produce 1000 Goose dies per wafer, with a die yield of 0.525. Each defect-free Goose chip makes a profit of $20. *How much **profit** will you make on each wafer of Goose chips ($)?*

✓ 10,500

Feedback

General Feedback

Profit = (dies per wafer)(die yield)(profit per defect-free chip)

We are given the number of complete dies per wafer is 1000. and the die yield is 0.525 (percentages of *defect-free* dies per wafer).

Thus the profit per wafer becomes (1000)(0.525)($20) = **$10,500**

6 4 / 4 points

The Testing Department has reported that our corporate server system has an FIT of 100. What is the MTTF (hours) for this system?

✓ 10 million

Feedback

General Feedback

Recall that FIT is rate of failures per billion (10^9) hours, and is the reciprocal of MTTF.

Thus, **MTTF = 10^9/FIT = 10^9/100 = 10^7 = 10 million hours**

7 4 / 4 points

We have a server system with a MTTF of 10 million hours. After failure, if it takes 10 days to get the system running again, what is the availability of the system?
Please show your answer to **6** significant digits after the decimal point.

✓ 0.999976

Feedback

General Feedback

It takes 10 days = 10 (24 hours/day) = 240 hours to get the system running again = MTTR

$$Availability = \frac{MTTF}{MTTF+MTTR}$$

$$= \frac{10^7}{10^7+240} = 0.999976 \ (rounded)$$

8 4 / 4 points

Your design team has developed a new processor with code name of "Peruna". This processor has an average cycles per instruction of 2. When run on a SPEC benchmark program that has 10^9 instructions, the total processor execution time is 10 seconds. What is the **clock rate** for this processor?

✓ 200 MHz

Feedback

General Feedback

Recall that CPU time = (instruction count)(CPI)(clock cycle time) = (instruction count)(CPI)/(clock rate)

We are given the following:

- CPU time = total processor execution time = 10 seconds
- CPI = average cycles per instruction = 2
- Instruction count for SPEC benchmark program = 10^9 instructions

We need to determine the clock rate. Using algebra, we revise the above equation as:

$$clock\ rate = \frac{(instruction\ count)(CPI)}{CPU\ time}$$
$$= \frac{(10^9\ instructions)\left(\frac{2\ cycles}{instruction}\right)}{10\ sec}$$
$$= 2\ x\ 10^8\ \frac{cycles}{sec} = 200\ MHz$$

9 4 / 4 points

Your design team has developed a new processor with code name of "Ranger". This processor has a clock cycle time of 2 ns and an average CPI of 1. The processor is tested on a SPEC benchmark program that has 2 x 10^9 instructions. What is the **execution time** for this program on this processor?

✓ 4 sec

Feedback

General Feedback

execution time = (instruction count)(CPI)(clock cycle time)

- We are given Instruction count (IC) = 2 x 10^9
- clock cycle time = 2 ns = 2 x 10^{-9} sec
- CPI (cycles/instruction) = 1

Thus,

$$execution\ time = (IC)\ (CPI)\ (clock\ cycle\ time)$$
$$= \left(2\ x\ 10^9\right)\ (1)\ \left(2\ x\ 10^{-9}\right) = 4\ sec$$

10 4 / 4 points

Your design team has designed a processor with code name of "Lyle". This processor has a clock cycle time of 3 ns. When the processor is run on a SPEC benchmark with 10^9 instructions, the resultant execution time is 6 seconds. What is the **average CPI** for this processor?

✓ 2

Feedback

General Feedback

Execution time = (instruction count)(CPI)(clock cycle time)

- We are given Instruction count (IC) = 10^9
- We are given clock cycle time = 3 ns = 3 x 10^{-9} sec
- Execution time is 6 sec

Thus,

$$CPI = \frac{Execution\ Time}{(IC)(clock\ cycle\ time)}$$
$$= \frac{6}{(10^9)(3\ x\ 10^{-9})} = 2$$

11 4 / 4 points

Your design team has designed a processor with code name of "Pony". This processor has a clock cycle time of 2 ns. When the processor is run on a SPEC benchmark with 10^9 instructions, the resultant execution time is 8 seconds. What is the **average CPI** for this processor?

✓ 4

Feedback

General Feedback

Execution time = (instruction count)(CPI)(clock cycle time)

- We are given Instruction count (IC) = 10^9
- We are given clock cycle time = 2 ns = 2 x 10^{-9} sec
- Execution time is 8 sec

Thus,

$$CPI = \frac{Execution\ Time}{(IC)(clock\ cycle\ time)}$$
$$= \frac{8}{(10^9)(2\ x\ 10^{-9})} = 4$$

12 4 / 4 points

We have a program that is 80% “parallelizable”: 80% of the program can be run in parallel, while 20% must be run sequentially. This program is currently run on a uniprocessor machine. What is the speedup if we run this program on a machine with 4 processors (quad-core machine)?

✓ 2.5

Feedback

General Feedback

Amdahl's Law, applied to this instance:

Speedup = $\frac{1}{(1-F)+\frac{F}{N}}$

F = fraction parallelizable = 80% or 0.8, N = amount of improvement = # cores = 4

Speedup = $\frac{1}{(1-0.8)+\frac{0.8}{4}}$ = 2.50 (rounded)

13 4 / 4 points

We have a program that is 80% “parallelizable”: 80% of the program can be run in parallel, while 20% must be run sequentially. This program is currently run on a uniprocessor machine. What is the speedup if we run this program on a machine with 2 processors (dual-core machine)?

✓ 1.67

Feedback

General Feedback

Amdahl's Law, applied to this instance:

Speedup = $\frac{1}{(1-F)+\frac{F}{N}}$

F = fraction parallelizable = 80% or 0.8, N = amount of improvement = # cores = 2

$$\text{Speedup} = \frac{1}{(1-F)+\frac{F}{S}}$$
$$= \frac{1}{(1-0.8)+\frac{0.8}{2}}$$
$$= \frac{1}{0.2+0.4} = \frac{1}{0.6} \approx 1.67$$

14 4 / 4 points

You are given the MIPS assembly code instruction **lw $s0, 16($s1)**. The corresponding machine code for this instruction (in hexadecimal form) is:

- AE30 0010
- 8E11 0010
- AE11 0010
- ✓ 8E30 0010

Feedback

General Feedback

Using MIPS Reference Sheet, we see that the **lw** instruction uses the I-format:

6 bits	5 bits	5 bits	16 bits
opcode	rs	rt	offset

Opcode for lw is 23_{16} = $10\ 0011_2$

Also from MIPS Reference Sheet, lw operation is R[rt] = M[R[rs]+SignExtImm]

Thus, rt = $s0 = register 16 = $1\ 0000_2$

and rs = $s1 = register 17 = $1\ 0001_2$

SignExtImm = offset = 16 = $0000\ 0000\ 0001\ 0000_2$

Filling in the I-format fields with the bit patterns gives the **binary representation**:

6 bits	5 bits	5 bits	16 bits
opcode	rs	rt	offset
10 0011	1 0001	1 0000	0000 0000 0001 0000

Or: $1000\ 1110\ 0011\ 0000\ 0000\ 0000\ 0001\ 0000_2$

To get the **hexadecimal form**, convert each set of 4 bits to its hex equivalent:

1000	1110	0011	0000	0000	0000	0001	0000
8	E	3	0	0	0	1	0

Or:

8E30 0010

15 4 points possible

You are given the following machine code instruction for MIPS in hexadecimal form: **0x21090005**. What is the corresponding **MIPS instruction** (assembly language form)?

Solutions to Questions 15 and 17 are at the end of this document

Waiting for grade

16 4 / 4 points

You are given the MIPS assembly code instruction **add $s0, $s1, $s2**. The corresponding machine code for this instruction (in hexadecimal form) is:

✓ 0232 8020

- 0211 9020
- 0211 9014
- 0232 8014

Feedback

General Feedback

From MIPS Reference Data sheet, the **add** instruction has an opcode of 0 and funct of 0x20

Operation of **add** instruction is R[rd] = R[rs] + R[rt]

This is an R-type instruction – format is:

op	rs	rt	rd	shamt	funct
6 bits	5	5	5	5	6

Op = 000000_2 and funct = 20H = 100000_2

There no shifting in an **add** instruction, so shamt = 0 = 00000_2

Register rd = $s0 = register 16 = 10000_2

Register rs = $s1 = register 17 = 10001_2

Register rt = $s2 = register 18 = 10010_2

Combine bit fields: op rs rt rd shamt funct => 000000 10001 10010 10000 00000 100000

Put in groups of 4 bits so that we can easily convert to hex:

$0000\ 0010\ 0011\ 0010\ 1000\ 0000\ 0010\ 0000_2 = 02328020_{16}$

17 4 points possible

Assume that C variables are assigned to MIPS registers as follows:

f = $s0, g = $s1, base address of B[0] stored in $s2

What is the corresponding MIPS code for the following C statement?

f = B[5] - g;

Solutions to Questions 15 and 17 are at the end of this document

Waiting for grade

18 4 / 4 points

Assume we have the following MIPS code, starting at memory address **2000H.**

If we use PC relative addressing, what is the **constant** for **DONE** in the **beq** instruction?

```
2000H   LOOP:   lw    $t2, 48($t1)
2004H           beq   $t0, $zero, DONE
2008H           add   $s1, $s1, $t2
200CH           addi  $t1, $t1, 4
2010H           addi  $t0, $t0, -1
2014H           j     LOOP
2018H   DONE:
```

✓ 4

Feedback

General Feedback

beq instruction is located at address 2004H in memory. After it is fetched, the program counter is incremented by 4 (PC = 2004H + 4H = 2008H)

Target address = PC + 4(offset), where **offset** is the **constant** in the **beq** instruction

Target address = 2018H (address of DONE label)

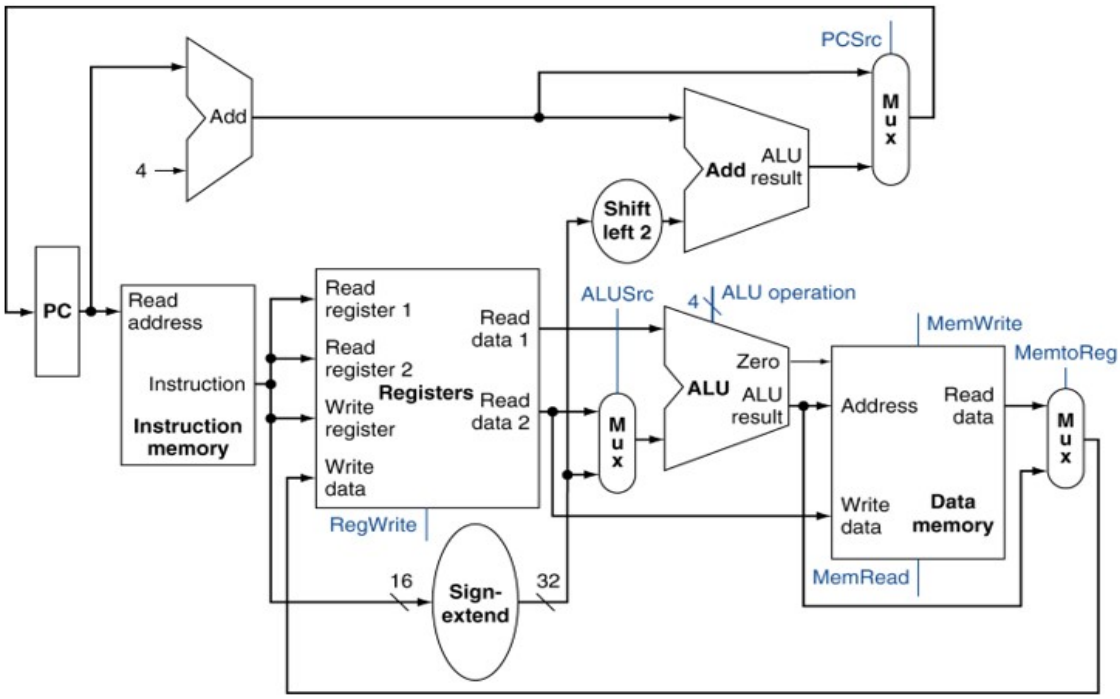
PC = 2008H

So, offset = (target address – PC) / 4 = (2018H – 2008H)/4

2018H - 2008H = 18H - 8H = 10H = 16

Offset = 16/4 = 4

19 4 / 4 points



The MIPS Datapath is executing the instruction **addi $s0, $s1, 5**

The control signals for this instruction are the following (fill in 0 or 1 for each value):

ALUSrc = ✓ 1 MemRead = ✓ 0 MemWrite = ✓ 0 MemtoReg = ✓ 0

Feedback

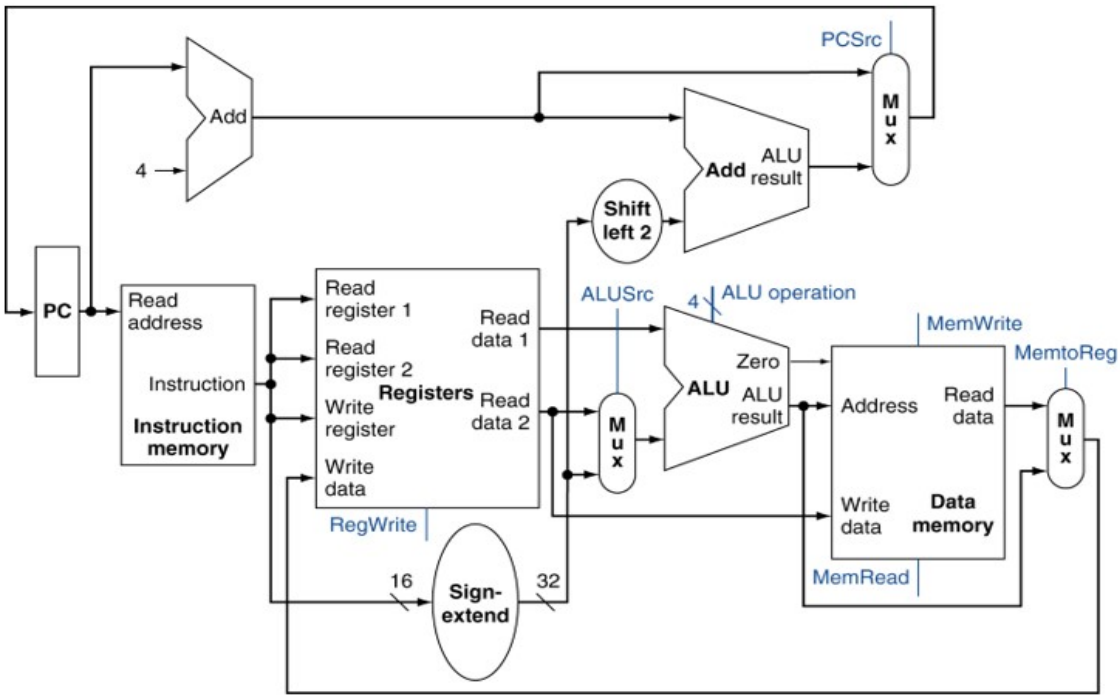
General Feedback

addi $s0, $s1, 5

Control Signal	Value	Comments
ALUSrc	1	Second ALU operand is offset
MemRead	0	No data memory reads
MemWrite	0	No data memory writes

MemtoReg	0	Register Write comes back from ALU result

20 4 / 4 points



The MIPS Datapath is executing the instruction **sub $s0, $s1, $s2**

The control signals for this instruction are the following (fill in 0 or 1 for each value):

ALUSrc = ✓ 0 MemRead = ✓ 0 MemWrite = ✓ 0 MemtoReg = ✓ 0

Feedback

General Feedback

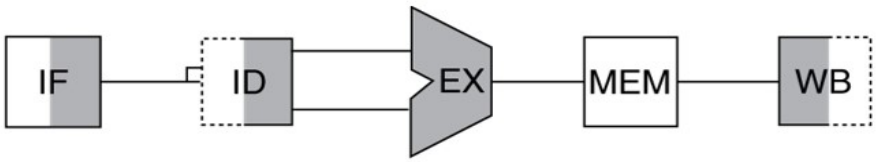
sub $s0, $s1, $s2

Control Signal	Value	Comments

ALUSrc	0	Second ALU operand is Rt
MemRead	0	No data memory reads
MemWrite	0	No data memory writes
MemtoReg	0	Register Write comes back from ALU result

21 4 / 4 points

We are using our MIPS pipeline with the five stages as shown below:



We are also given the following instruction sequence for our 5-stage MIPS pipeline:

```
LW   R2, 16(R1)
ADD  R3, R5, R2
SUB  R6, R5, R4
SW   R6, 16(R1)
```

Identify the registers that have **data hazards** (check all that apply):

- [ ] R4
- [ ] R5
- [ ] R1
- [ ] R0
- [x] R2
- [x] R6
- [ ] R3

Feedback

General Feedback

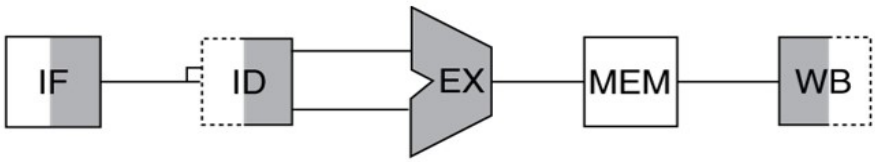
```
LW   R2, 16(R1)

ADD  R3, R5, R2

SUB  R6, R5, R4

SW   R6, 16(R1)
```

"LW" produces result in R2 that is input to "ADD", so register R2 has a data hazard

"SUB" produces result in R6 that is input to "SW", so register R6 has a data hazard

22 4 / 4 points

We are using our MIPS pipeline with the five stages as shown below:



We are also given the following instruction sequence for our 5-stage MIPS pipeline:

```
ADD R0, R2, R1
OR  R4, R0, R3
SUB R5, R0, R2
AND R6, R4, R3
```

Identify the registers that have **data hazards** (check all that apply):

- [ ] R6
- [ ] R2
- [ ] R1
- [x] R4
- [ ] R3
- [ ] R5

✓ ☑ R0

Feedback

General Feedback

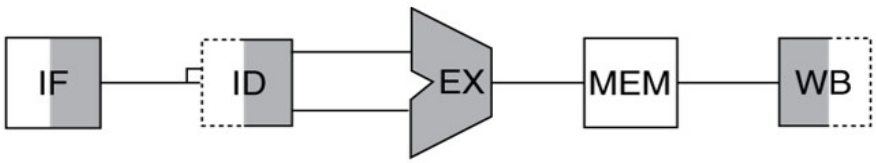
```
ADD  R0, R2, R1

OR   R4, R0, R3

SUB  R5, R0, R2

AND  R6, R4, R3
```

"ADD" produces result in R0 that is input to "OR" and "SUB", so register R0 has a data hazard

"OR" produces result in R4 that is input to "AND", so register R4 has a data hazard

23 4 / 4 points

We are using our MIPS pipeline with the five stages as shown below:



We are also given the following instruction sequence for our 5-stage MIPS pipeline:

```
ADD  R5, R2, R1
SW   R5, 32(R1)
SUB  R3, R5, R0
```

How many stalls are required after the SW instruction?

☐ 1

✓ ☑ 0

- [ ] 2
- [ ] 3
- [ ] 4

Feedback

General Feedback

SW uses register R5 as an input register during its ID stage (CC 3), but does not change the value of R5.

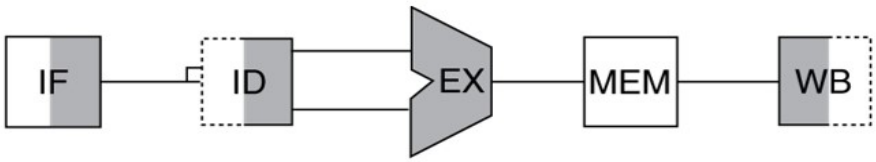
SUB also uses register R5 as an input. However, since SW does not change the value of R5, there is no data hazard.

Therefore, **0** stalls are required.

---

24 4 / 4 points

We are using our MIPS pipeline with the five stages as shown below:



We are also given the following instruction sequence for our 5-stage MIPS pipeline:

```
ADD  R5, R2, R1
SW   R5, 32(R1)
SUB  R3, R5, R0
```

How many stalls are required after the ADD instruction?

- [ ] 4
- [ ] 3
- [x] 2
- [ ] 1
- [ ] 0

Feedback

General Feedback

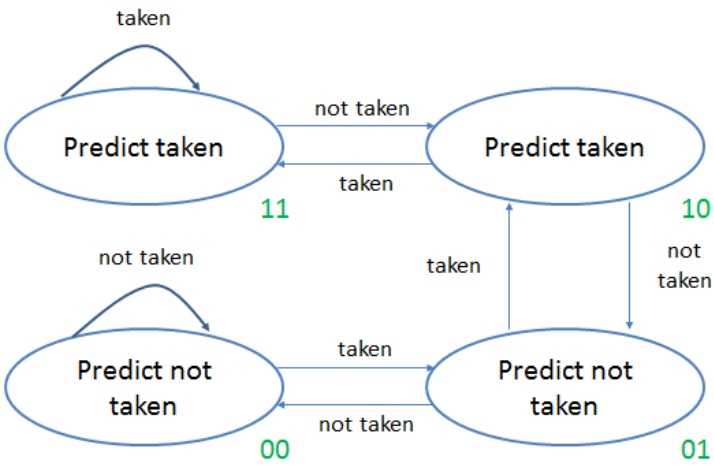
There is a data hazard in register R5

Clock Cycle (CC)	1	2	3	4	5	6
ADD R5, R2, R1	IF	ID	EX	MEM	WB	
SW R5, 32(R1)		IF	ID	EX	MEM	WB

SW uses register R5 as an input register during its ID stage (CC 3), but R5 is not updated until ADDIs WB stage (CC 5). Therefore, we need to stall the SW instruction by **2** cycles.

25 4 / 4 points

We have a sequence of branch outcomes, where T = branch taken, and NT = branch not taken. What is the **accuracy** of the **two-bit branch predictor** for the following sequence, assuming that the predictor starts in state **11**?
Sequence is: **NT, T, T, T, T**



✓ 80%

Feedback

General Feedback

We are starting in the the upper left state (11), so we follow the sequence as

Current state Prediction Branch outcome Accurate? Next state

11	T	NT	No	10
10	T	T	Yes	11
11	T	T	Yes	11
11	T	T	Yes	11
11	T	T	Yes	11

So the 2-bit predictor is accurate 4/5 time = **80%** for this sequence

Question 15 Solution

You are given the following machine code instruction for MIPS in hexadecimal form: **0x21090005**. What is the corresponding **MIPS instruction** (assembly language form)?

SOLUTION:

0x21090005 – convert to binary:

0010 0001 0000 1001 0000 0000 0000 0101

Recall that opcode is first 6 bits of string.

0010 0001 0000 1001 0000 0000 0000 0101

First 6 bits are 0010 00 = 001000_2, so opcode = 8

From MIPS Reference Data sheet, opcode 8 is the **addi** instruction

Operation of **addi** instruction is R[rt] = R[rs] + SignExtImm

This is an I-type instruction - format is:

op	rs	rt	constant/address
6 bits	5 bits	5 bits	16 bits

So our bit string is split into the fields as follows:

op	rs	rt	constant/address
6 bits	5 bits	5 bits	16 bits
001000	01000	01001	0000 0000 0000 0101

Note that the register fields are the following:

Register rs = 01000_2 = 8 = register $t0

Register rt = 01001_2 = 9 = register $t1

The constant is 0000 0000 0000 0101_2 = 5

Thus, the MIPS instruction is **addi $t1, $t0, 5**

Question 17 Solution

Assume that C variables are assigned to MIPS registers as follows:

f = $s0, g = $s1, base address of B[0] stored in $s2

What is the corresponding MIPS code for the following C statement?

```
f = B[5] - g;
```

SOLUTION:

```
f = B[5] - g;
```

Variable f = register $s0, variable g = register $s1

Base address for array B is stored in $s2 (location of B[0])

Offset of B[5] from base address B[0] is 5*4 = 20 (since 4 bytes per array element)

MIPS assembly code is:

```
lw    $t0,20($s2)           # load B[5] into temp register
sub   $s0,$t0,$s1           # f = B[5] - g
```