جامعة بيروت العربية
BEIRUT ARAB UNIVERSITY

Date: 14-12-2021
Duration: 110 minutes
Mark: /60
No. of pages including cover:

Final Exam

Semester: Fall 2021/2022

Faculty : Faculty of Science
Department : Math & Computer Science
Division/ program: Computer Science
Course Name : Computer Organization
Course Code: CMPS 343

Student's Name: ..... May ..... **ID:** .....
Section/ Group: ..... **Seat Number:** .....

INSTRUCTIONS:
1- Any kind of cheating will subject the student to the penalties specified by the University rules
2- Exam is open-notes

Question	Mark	Out of
Obj.	10	
One	8	
Two	6	
Three	4	
Four	6	
Five	12	
Six	8	
Seven	6	
Total	60	

Total marks in letters

Examiner's Name: Dr. May Itani

Signature: .....

Objective Part – 10 pts

Choose only one answer else no grade will be given. Problems that include calculation show your work,

Q1. `lh` is a ?

Load halfword

`lh rt, address`

0x21	rs	rt	Offset
6	5	5	16

a. R- type instruction
b. I - type instruction
c. J- type instruction

Q2. Noting that IR=Instruction Register, MBR=Memory Buffer Register, MAR=Memory Address Register, D0=Data Register, PC = Program Counter, SR = Status Register, A0= Address Register, then the fetch-decode-execute cycle involves the following:

a. IR, MBR, MAR, D0 and PC
b. IR, PC, MBR and MAR
c. SR, PC, MAR and MBR
d. A0,D0,IR and PC

Q3. A locality principle is that

a. Variables are mostly local in any program
b. Program cannot run without local variables
c. Memory references tend to use small fraction of memory locations
d. Memory references are local to the program

Q4. A data path is

a. Path of the data on the main computer bus
b. Path of the data between CPU and main memory
c. Buses inside computers that contain data
d. ALU and its input and output

Q5. A cache hit happens when

a. Main Memory and hard disk are full
b. Cache memory is used by huge number of computers
c. Cache is hit by interrupt request on memory bus
d. Memory word is found on cache

Q6. State whether each of the following is true and/or false

i. Register-register operations are faster than memory-register operations----------

j. Assembly language is a readable representation of machine language----------

k. Main reason for using cache memory is that it is cheaper than main memory---------- False

l. A pipe-lining is a mechanism of processor-level parallelism---------- overlapping

m. Instruction Set Architecture has to do with the data path cycle in the CPU ----------

Subjective Part

Question 1 –8 pts.

Convert the following high level fragment to equivalent MIPS assembly language.

- Steps:
  - Allocate registers to program variables (use convention for method arguments)
  - Produce code for the body of the function.
  - Preserve registers across the procedure call.

```
swap (int v[], int k)

{ int temp;
  temp=v[k];
  v[k]=v[k+1];
  v[k+1]=temp;
}
```

let v ≡ $a0 base address
k ≡ $a1

```
swap :
    sll  $t1, $a1, 2     # t1 = 4*k
    add  $t1, $a0, $t1   # t1 has address of v[k]
    lw   $t0, 0($t1)     # temp = v[k]
    lw   $t2, 4($t1)     # t2 = v[k+1]
    sw   $t2, 0($t1)     # v[k] = $t2
    sw   $t0, 4($t1)     # v[k+1] = temp
    jr   $ra             # return to caller
```

Question 2—6pts. (8)

In this part we examine how pipelining affects the clock cycle time of the processor. Assume individual stages of the data path have the following latencies:

IF	ID	EX	MEM	WB
200 ps	150 ps	120 ps	190 ps	140 ps

a. What is the **clock cycle time** in a single cycle(non-pipelined) processor? Show steps in details.

same as latency of lw longest instruction.

$CC_{time} = 200 + 150 + 120 + 190 + 140 = 800ps$

b. What is the **clock cycle time** in a pipelined processor? Show steps in details.

longest stage (phase)

$CC_{time} = 200ps$ (pipelined)

c. What is the **latency** of a **sw** instruction in a **non-pipelined(single cycle)** processor? Show steps in details.

sw has 4 stages

IF, ID, EX, MEM

$latency_{sw} = 200 + 150 + 120 + 190 = 660ps$

d. What is the latency of **`branch`** instruction in a **pipelined** processor? Show steps in details.

latency of branch = 3 x 200

(neglecting NOPs) = 600ps

with NOPs : All take 5 stages = 1000ps.

Question 3—4pts.

Assume the distribution of instructions that run on the processor is:

ALU	45%
BEQ	10%
LW	20%
SW	25%

What are the hardware functional units used by the branch instruction?

I-MEM
Register File
ALU

What is the utilization of the register block?

100% utilization
All instructions use Register block.

What is the utilization of data memory?

lw & sw use data Mem
20 + 25 = 45% utilization

Question 4—6pts.

Given the following sequence of instructions:

```
1  lw $s2, 0($s1)
2  lw $s1, 40($s6)
3  sub $s6, $s1, $s2
4  add $s6, $s2, $s2
5  or $s3, $s6, $zero
6  sw $s6, 50($s1)
```

I3 depends on I1 ($s2)
I3 depends on I2 ($s1)
I4 depends on I1 ($s2)
I5 " " I4 ($s6)

A data dependence is a dependence of one instruction B on another instruction A because the value produced by A is read by B.

List all data dependencies. When listing data dependencies, you need to mention A, B, and the location (in this case, the register that causes the dependence).

I6 depends on I2 ($s1)
I6 depends on I4 ($s6)

I 3 depends on I 1 ($S2)
I 3 depends on I 2 ($S1)

BACK TO PREVIOUS PAGE

For the first three instructions in the sequence above, complete the pipeline diagram below (instructions on the left, cycles on top). Insert the characters IF, ID, EX, MEM, WB for each instruction in the boxes.

Label all data stalls (Draw an X in the box).

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15
I1	IF	ID	EX	MEM	WB										
I2		IF	ID	EX	MEM	WB									
I3			IF	X	X	ID	EX	NOP	WB						

10

Question 5—10pts.

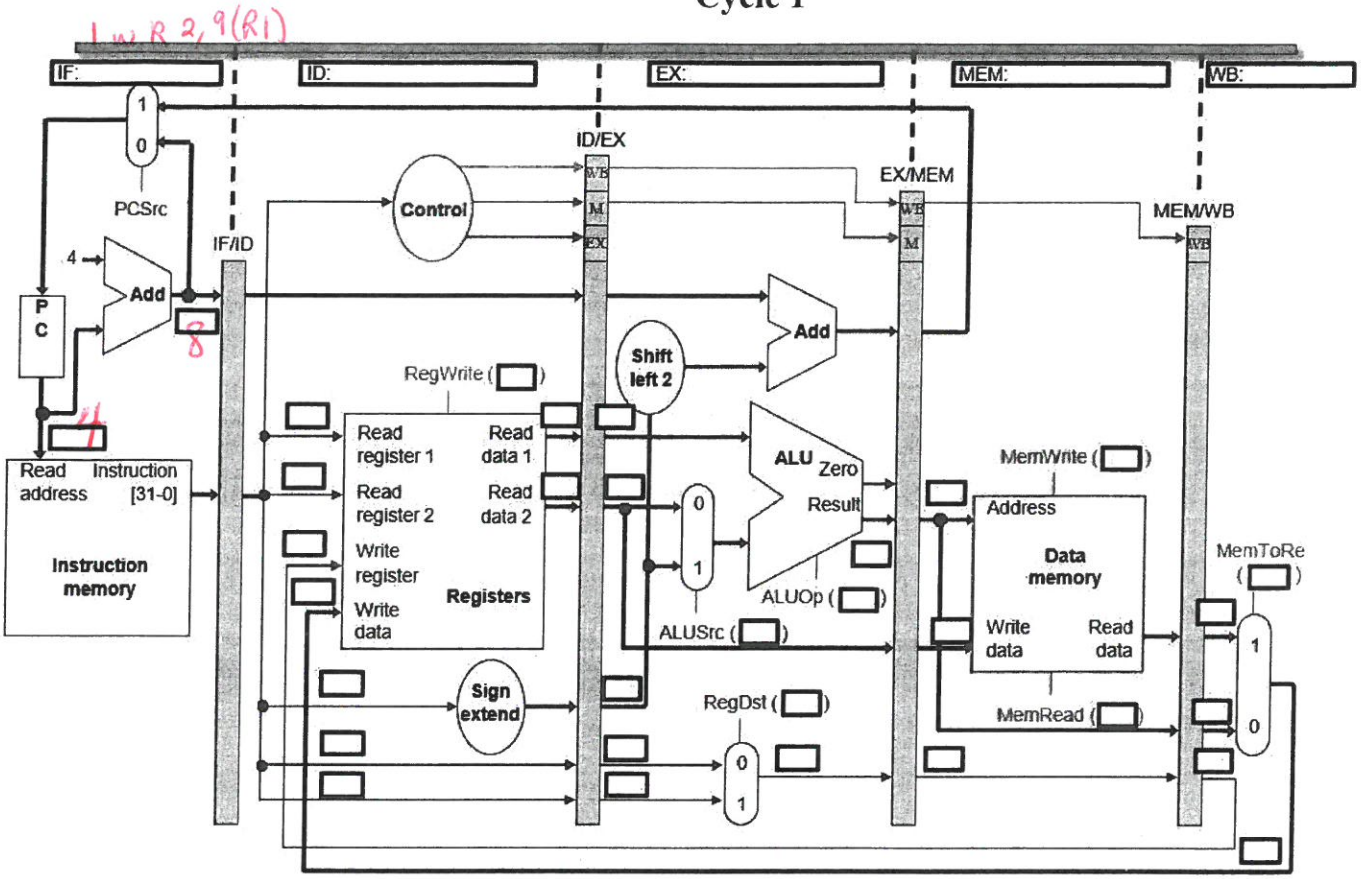
Assume that data memory is all zeros and that the processor's registers have the following values before execution of the given sequence of instructions.

R1	R2	R3	R4	R5	R6	R7
2	4	6	8	5	6	7

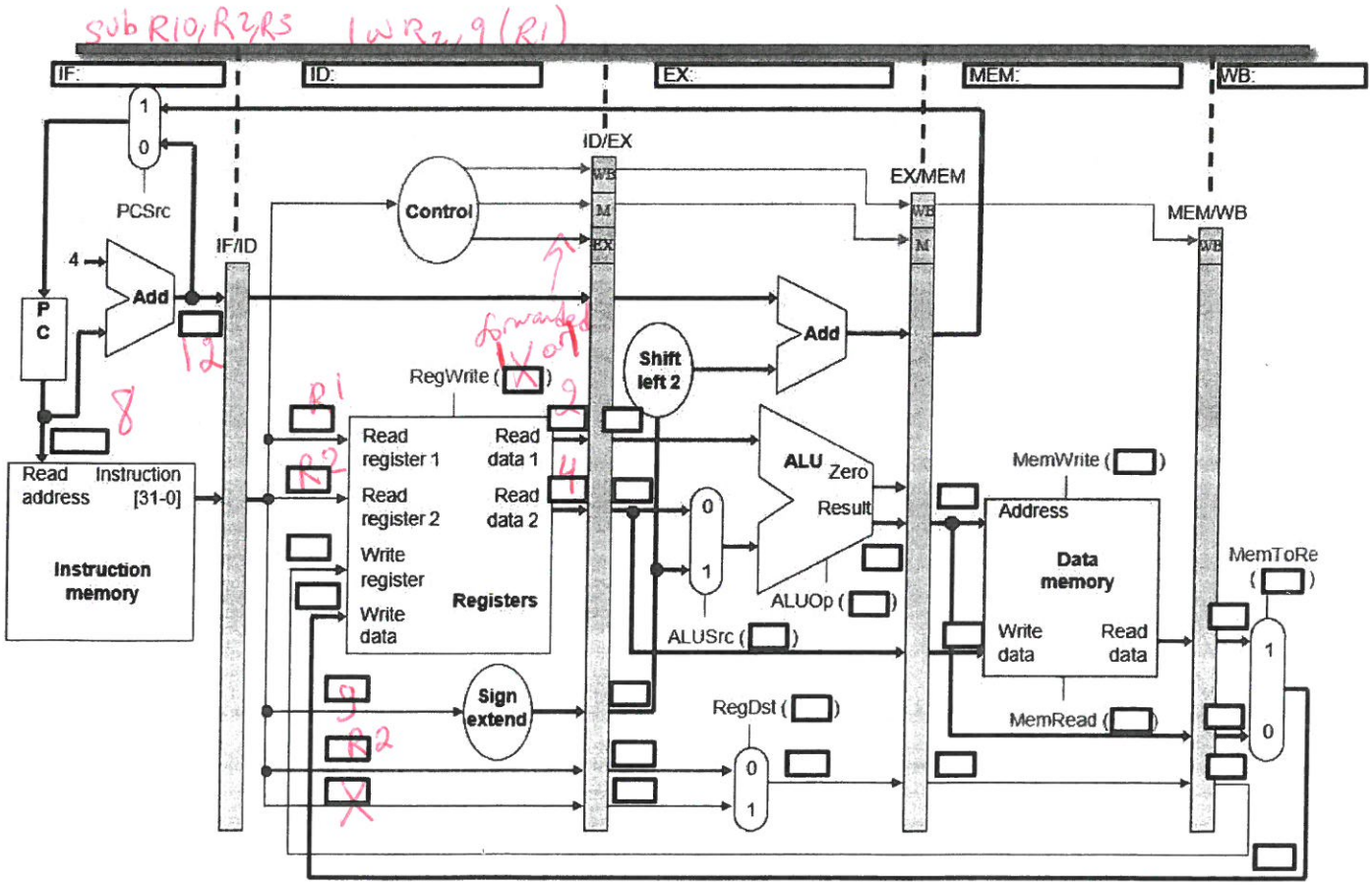
Execute the following instructions by filling the empty box ☐ with relevant values that will be set/generated during execution in the first four cycles of a pipelined processor. Assume instruction addresses are given in decimal notation.

```
 4: lw R2, 9(R1)

 8: sub R10, R2, R3

12: and R11, R4, R5

16: or R12, R6, R7
```

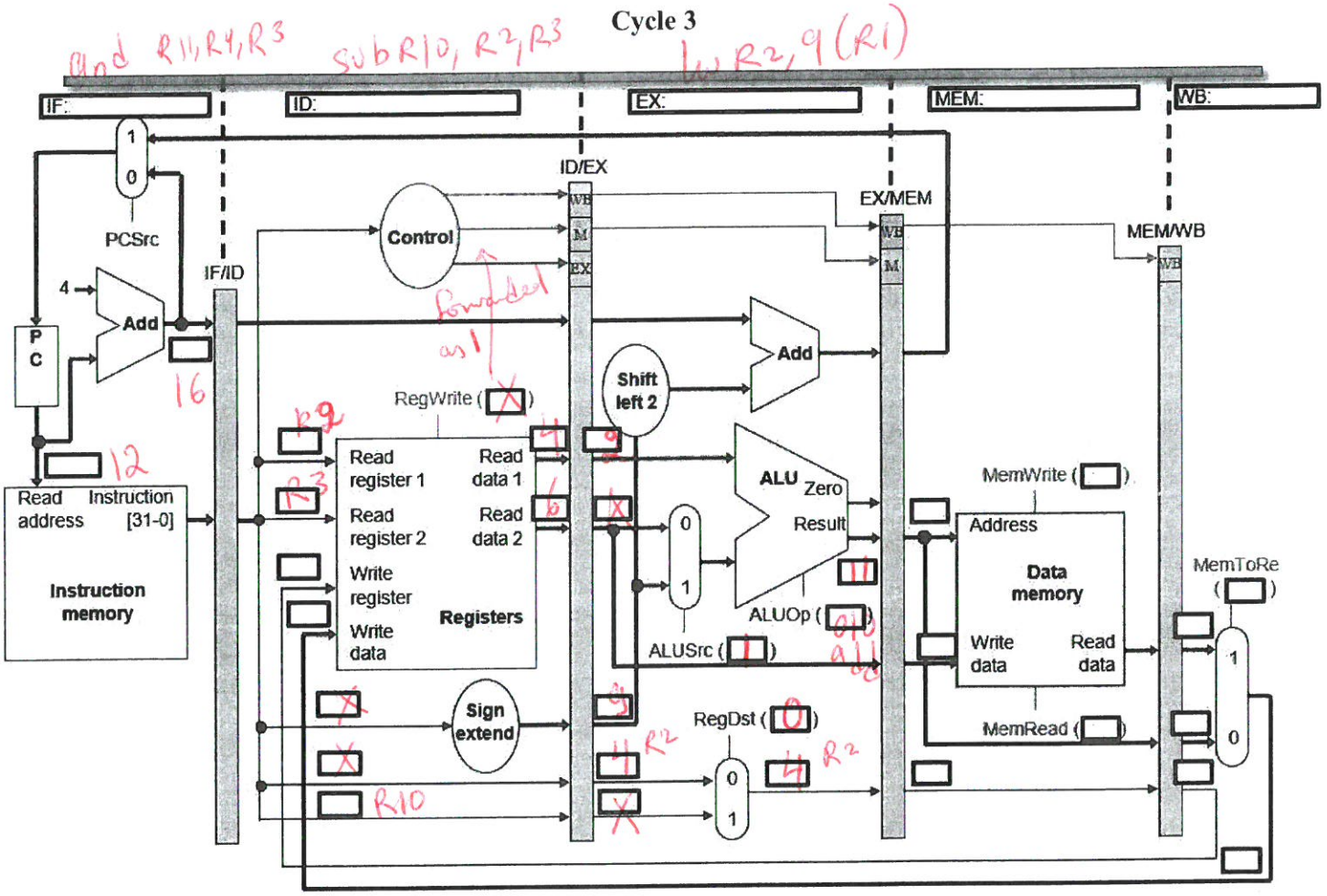
Cycle 1



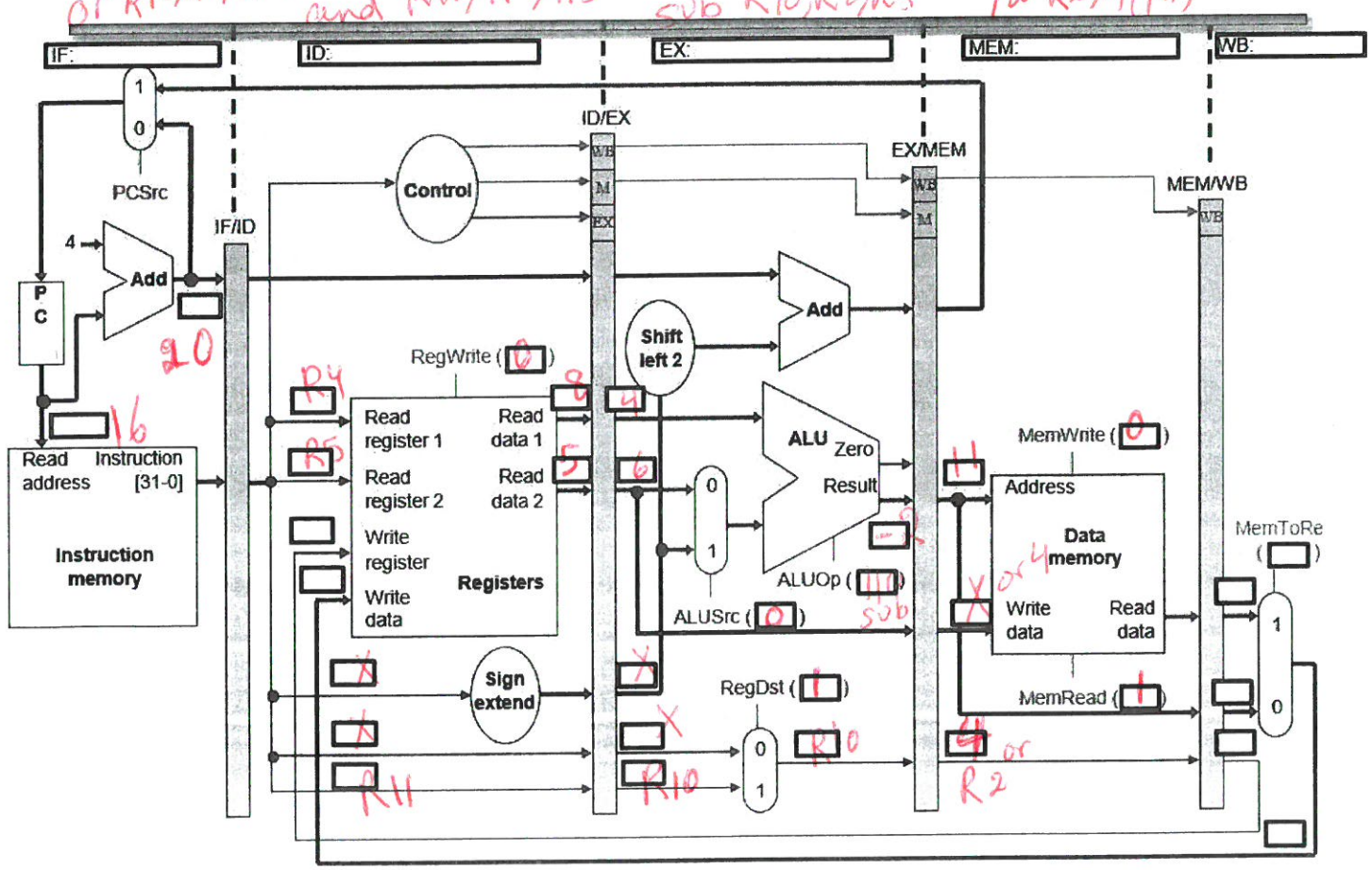
Cycle 2



Cycle 3



Cycle 4



Question 6—8pts.

For this question, you are given a 16-byte cache (initially empty).
For each of the given cache configurations show the result of memory accesses (sequence of hits and misses) given the following sequence of block addresses: 2, 8, 2, 6, 5, 8

1. 2-byte blocks, direct mapped (8 blocks)

Block Address	Cache Block
2	2 mod 8 = 2
5	5 mod 8 = 5
6	6 mod 8 = 6
8	8 mod 8 = 0

Address of memory block accessed	
2	miss
8	miss
2	hit
6	miss
5	miss
8	hit

2. 2-byte blocks (Cache has 8 blocks), 2-way set associative

(each set has 2 blocks) ⇒ 4 sets

Block Address	Cache Set
2	2 mod 4 = 2
5	5 mod 4 = 1
6	6 mod 4 = 2
8	8 mod 4 = 0

mod 4

Address of memory block accessed	
2	miss
8	miss
2	hit
6	miss
5	miss
8	hit

3. 2-byte blocks, fully associative (8 blocks)

Address of memory block accessed	
2	miss
8	miss
2	hit
6	miss
5	miss
8	hit

Question 7—6pts.

Consider the following C++ code.

```
void guess(int arr[ ], int n)
{
for(int i=0 ; i<n ; i=i+1)
arr[i] = i+1;
}
```

Assume you generated MIPS assembly code for the C++ code using two different compilers. Compiler 1 generated the code in the first column while compiler 2 generated the code in the second column.

Compiler 1		Compiler 2	
	add $t0, $zero, $zero		sll $a1, $a1, 2
	add $t1, $zero, $zero		sll $t3, $zero, 2
loop:	bge $t1, $a1, back		sll $t1, $zero, 2
	sll $t2, $t1, 2		bge $t1, $a1, back
	add $t2, $t2, $a0	loop:	add $t2, $t1, $a0
	addi $t3, $t1, 1		addi $t3, $t3, 1
	sw $t3, 0($t2)		sw $t3, 0($t2)
	addi $t1, $t1, 1		addi $t1, $t1, 4
	j loop		blt $t1, $a1 loop
back:	jal $ra	back:	jal $ra

Assume **arithmetic/logic instructions require 2 cycles** to execute, **loads 5 cycles, branches 3 cycles, and jumps 1 cycle.**

Fill the table below with the number of instructions of each type that will be executed by the code generated by compiler 1 and by compiler 2 when the function guess(arr, 100) is invoked.

Instruction Type	Compiler 1 Code	Compiler 2 Code
Arithmetic/logic	402	303
Loads	100	100
Branches	100	101
Jumps	101	1

Compute the average number of clock cycles per instruction (CPI) for each version of the program

Average CPI compiler 1: $(402 \times 2 + 100 \times 5 + 101 \times 3 + 101 \times 1)/703$

$1708/703 = 2,429$

Average CPI compiler 2: $(303 \times 2 + 100 \times 5 + 101 \times 3 + 1 \times 1)/505$

$1410/505 \quad 2,79$

Which version of guess() is faster if you run it on a 1 GHz processor and by how much?

M_2 better $1708/1410$

$1708/1412 = 1,209$ 121%

compiler 2 better

$P_{M2}/P_{M1} = CPI_{M1}/CPI_{M2}$