

組別： _____ 簽名： _____

[group 1]

1. The following instructions are I-type, R-type, or J-type. Classify each one.

1. add \$t0, \$s3, \$s4
2. lw \$t1, 8(\$s2)
3. beq \$t0, \$t1, Label
4. j Loop
5. slt \$t2, \$s3, \$s4
6. addi \$s0, \$s0, 1
7. sltu \$t3, \$s5, \$s6
8. sw \$t0, 0(\$s1)

Ans: 1. R 2. I 3. I 4. J 5. R 6. I 7. R 8. I

[group 12]

2. Given the following binary encoding of a MIPS instruction:

op rs rt rd shamt funct
 000000 00000 01100 10011 01010 000010

Convert it into the corresponding Assembly instruction.

Hint: (rt = $01100_2 = 12 = \$t4$) (rd = $10011_2 = 19 = \$s3$) (funct = $000010_2 = 2 \rightarrow \text{srl}$)

Ans:

op = 000000 $\rightarrow$ R-type

funct = $000010_2 = 2 \rightarrow \text{srl}$ (shift right logical)

rs = 00000 $\rightarrow$ fixed to 0 for shift instructions

rt = $01100_2 = 12 = \$t4$, rd = $10011_2 = 19 = \$s3$, shamt = $01010_2 = 10$

Therefore, the corresponding Assembly instruction is: srl \$s3, \$t4, 10

[group 3]

3. As a reference example, encoding the instruction add \$t0, \$s1, \$s2 by hand yields: opcode=0, rs=17, rt=18, rd=8, shamt=0, funct=32, giving 0x02324020.

Using the same register assignment (rs=\$s1, rt=\$s2, rd=\$t0), give the R-format fields and 32-bit binary encoding for slt \$t0, \$s1, \$s2, then convert to hex.

Ans:

Field	opcode	rs	rt	rd	shamt	funct
Decimal	0	17	18	8	0	42
Binary	000000	10001	10010	01000	00000	101010

Full binary: 00000010001100100100000000101010

Hex: 0x0232402A

[group 13]

4. Assume that the opcodes of the add, addi and lw instructions are respectively 000000, 001000 and 100011 in binary notation. Translate the following MIPS machine codes into corresponding assembly codes.

(1) 0000000010010101001000000000100000

(2) 00100010110101011111111111001110

(3) 100011010010100000000010010110000

Ans:

(1) add \$t0, \$t1, \$t2

(2) addi \$s5, \$s6, -50

(3) lw \$t0, 1200(\$t1)

[group 2]

5. Suppose a 32-bit register \$t0 holds the following hexadecimal value: \$t0 = FFFFFFFE5 and the MIPS assembly instruction sequence of sra \$t1, \$t0, 3. What is the signed decimal value does \$t1 represent?

Ans:

FFFFFFFE5 = 1111 1111 1111 1111 1111 1111 1110 0101

SRA --> 1111 1111 1111 1111 1111 1111 1111 1100

decimal value:

0000 0000 0000 0000 0000 0000 0000 0011

add 1 --> 0000 0000 0000 0000 0000 0000 0000 0100

answer: -4

[group 7]

6. addi \$t0, \$s0, 5

sll \$t1, \$t0, 2

srl \$t2, \$t1, 1

Given that the final value of \$t2 is 184, determine the original value stored in \$s0. Assume all intermediate values fit within 32 bits and no overflow occurs.

Ans:

\$t2 = 184

Before srl by 1: \$t1 = $184 \times 2 = 368$

Before sll by 2: \$t0 = $368 \div 4 = 92$

Before addi 5: \$s0 = $92 - 5 = 87$

So, \$s0 = 87

[group 4]

7. Please write a simple MIPS code to double a number if it is odd (not a multiple of 2).
Number input is stored at \$s0.
Save the result directly to \$s0.
Hint: You may use shift, logical immediate, and branch instructions.

Ans:

Reference Answer:

A number is odd if a number's LSB is 1.

Therefore, have to AND the input number with 1, and check if the result is 1.

If it is 1, bitwise shift left by 1 makes it double.

```
andi $t0, $s0, 1    # Check LSB: $t0 = $s0 & 1
beq $t0, $zero, Exit # Jump to Exit if it's even ($t0 == 0)
sll $s0, $s0, 1     # else $s0 = $s0 << 1
```

Exit:

[group 5]

8. (a) Translate into MIPS, with f, g, h, i, j in \$s0–\$s4:
if (i == j) f = g + h;
(b) The C code tests for equality, yet the compiled code normally begins with bne. Explain why.

Ans:

- (a)
- ```
bne $s3, $s4, Exit # if (i != j) skip the then-part
add $s0, $s1, $s2 # f = g + h
```

Exit:

- (b) Assembly branches jump away, while the "then" block must fall through in sequence. Testing the opposite condition lets the code branch past the then-part when the condition fails and simply continue when it succeeds — otherwise an extra unconditional jump would be needed.

[group 8]

9. Show how to implement the following comparisons using slt and conditional branches.
1.  $a < b$
  2.  $a \leq b$
  3.  $a > b$
  4.  $a \geq b$

Ans:

Assume:  $a = \$s0$ ,  $b = \$s1$

|                        |                             |                           |                                |
|------------------------|-----------------------------|---------------------------|--------------------------------|
| 1. $a < b$             | 2. $a \leq b$               | 3. $a > b$                | 4. $a \geq b$                  |
| slt \$t0, \$s0, \$s1   | slt \$t0, \$s1, \$s0        | slt \$t0, \$s1, \$s0      | slt \$t0, \$s0, \$s1           |
| bne \$t0, \$zero, Less | beq \$t0, \$zero, LessEqual | bne \$t0, \$zero, Greater | beq \$t0, \$zero, GreaterEqual |

Explanation:

slt \$t0, \$s0, \$s1 sets \$t0 to 1 if  $a < b$ , and 0 otherwise.

Therefore:

$a < b \rightarrow \text{slt}(a, b)$

$a > b \rightarrow \text{slt}(b, a)$

$a \leq b \rightarrow \text{NOT}(b < a)$

$a \geq b \rightarrow \text{NOT}(a < b)$

[group 10]

10. Assume \$t0 holds the value 0xFFFFF000. What is the value of \$t2 after the following instructions?

```
slt $t2, $t0, $0
beq $t2, $0, ELSE
addi $t2, $t2, 3
j DONE
ELSE: addi $t2, $t2, 5
DONE:
```

Ans:

\$t0 = 0xFFFFF000 is negative (signed)  
slt \$t2, \$t0, \$0 → \$t0 < 0 is true → \$t2 = 1  
beq \$t2, \$0, ELSE → 1 != 0, 不跳到 ELSE, 繼續下一行  
addi \$t2, \$t2, 3 → \$t2 = 1 + 3 = 4  
j DONE → 直接跳到 DONE  
所以答案 \$t2 = 4

[group 14]

11. Given a MIPS CPU for running a program. Assume the PC (program counter) currently contains address 0x00000040. Which of the following statements are correct ?

- a. It is possible to use a single beq to get to address 0x08000040
- b. It is possible to use a single beq to get to address 0xFFFFF000

Ans:

因為 branch ( I-type ) : [op 6] [rs 5] [rt 5] [constant 16]

所以 branch 的範圍由 [constant 16] 所控制 / 決定

Constant 範圍

Max : 0111 1111 1111 1111 :  $2^{15} - 1 = 32767$

Min : 1000 0000 0000 0000 :  $-2^{15} = -32768$

Offset 範圍

Max \* 4 = 0111 1111 1111 1111 00 = 01 1111 1111 1111 1100 = 1FFFC

Min \* 4 = 1000 0000 0000 0000 00 = 10 0000 0000 0000 0000 = -20000

所以 branch 最多能往下跳 1FFFC, 往上跳 20000

A. No

Current PC : 0x00000040, Target : 0x08000040

PC + 4 能往下跳的最遠 : 0x00000044 + 1FFFC = 0x00020040

B. Yes

Current PC : 0x00000040, Target : FFFFFFF00

PC + 4 能往回跳的最遠 : 0x00000044 - 20000 = FFFE0044