NATIONAL UNIVERSITY OF SINGAPORE

CS2100 – COMPUTER ORGANISATION

Midterm Assessment
AY2024/25 Semester 2

Time Allowed: 1.5 Hours

Instructions (please read carefully):

1. This assessment consists of **24** questions printed on **13** pages.
2. This is an **OPEN BOOK** assessment. Calculators are allowed but not any other electronic devices.
3. You are to write your answers on the single A4 Answer Sheet provided. We will collect ONLY your Answer Sheet at the end of the test. You may keep this question paper.
4. You are to shade your Student Number on the Answer Sheet correctly with a pencil. Please do this at the start of the assessment and double check that you have shaded it correctly.
5. **All** questions **must** be answered in the boxes provided on the Answer Sheet. Any writing outside the designated answer area will not be graded.
6. Do not attach any additional sheet of papers to the Answer Sheet, it will jam our scanning and it will be disregarded.
7. You are allowed to write all your answers with a pencil (2B or above), as long as it is legible. Marks may be deducted for illegible handwriting.
8. Page 13 contains the MIPS Reference Data Sheet.

Section-A: Multiple Choice Questions (20 x 1 = 20 marks)

Choose the best answer for each question and shade the corresponding bubble on the Answer Sheet with a pencil (2B or above). For correction, please erase the shading completely or the auto-grading software may detect multiple answers.

1. In a computer systems class, students were instructed to perform addition and subtraction using 1's complement arithmetic, with the assurance that no overflow would occur. However, one student accidentally treated the numbers as 2's complement, leading to an overflow. Identify the original equation that the student is most likely working on, which would not overflow in 1's complement but does in 2's complement.
    A. 0011 – 1010
    B. 0010 – 0101
    C. 1101 + 1011
    D. 1110 + 1001
    E. None of the above.

2. To ensure accurate transfer of characters encoded in standard ASCII, an odd parity bit scheme is used. An additional parity bit is added to the ASCII representation to maintain an odd number of 1s. Which of the following errors can this scheme detect?
    A. The character 'A' (65_{10}) accidentally transferred as 'P' (80_{10})
    B. The character '9' (57_{10}) accidentally transferred as 'w' (119_{10})
    C. The character 'L' (76_{10}) accidentally transferred as 'z' (122_{10})
    D. The character '+' (43_{10}) accidentally transferred as '}' (125_{10})
    E. None of the above.

3. Given a number system on n bits, where $n \geq 8$, determine how many distinct binary representations correspond to the same integer value in at least two of the following representation schemes: Sign-and-Magnitude, Two's Complement, and Excess-($2^{n-1}+2^{n-2}$).
    A. $2^{n-1} + 2$
    B. $2^{n-1} + 4$
    C. $2^{n-1} + 2^{n-3}$
    D. $2^{n-1} + 2^{n-2}$
    E. None of the above.

4. The number 1201 in base 5 is equivalent to 246 in which base system?
    A. Base-7
    B. Base-8
    C. Base-9
    D. Base-11
    E. None of the above.

5. Given the decimal number 123456789.2425. Suppose we convert it into its closest ternary (base 3), quaternary (base 4), and quinary (base 5) representations, with each having 3 digits in the fractional part. Which of the following statements is true if we refer to the answers as base-3 answer, base-4 answer and base-5 answer respectively?
    A. The base-3 answer is closest to the original decimal number.
    B. The base-4 answer is closest to the original decimal number.
    C. The base-5 answer is closest to the original decimal number.
    D. The base-4 and base-5 answers are equally closest to the original decimal number.
    E. None of the above.

6. What is the IEEE 754 single-precision (32-bit) representation of the decimal number **-12.375**?
    A. 0x41460000
    B. 0xC1460000
    C. 0x81C60000
    D. 0xC1630000
    E. None of the above.

7. What is the decimal value that is represented as **0x41A80000** in the IEEE 754 single-precision (32-bit) representation?
    A. 5
    B. 10.5
    C. 21
    D. 1.0101×2^{131}
    E. None of the above.

8. Consider the following MIPS code:

```
     addi $t0, $zero, -1  # Initialize $t0 with -1
     addi $t1, $zero, -1
loop:
     add  $t0, $t0, $t1
     bne  $t0, $0, loop   # Repeat until $t0 becomes 0
     j exit
exit:
```

How many times will the **add $t0, $t0, $t1** instruction be executed before the program exits?

A. 0
B. 1
C. 2147483648
D. 4294967295
E. None of the above.

9. Suppose a jump instruction "j xxx" is located at address **0x4FFFFFFC**, which of the following target addresses are **unreachable** by this jump instruction?

(i) 0x38888888
(ii) 0x4FFFFFFC
(iii) 0x50000000
(iv) 0x5FFFFFFC

A. Only (i).
B. Only (i) and (ii).
C. Only (i), (iii) and (iv).
D. All of (i), (ii), (iii) and (iv).
E. None of the above.

10. Consider the following C program, which rotates a 32-bit unsigned integer **v** to the right by 4 bits. Hence, the output of the program would be "y = 0x81234567".

```
#include <stdio.h>
int main() {
    unsigned int v = 0x12345678;
    unsigned int y = (v >> 4) | (v << 28);
    printf("y = 0x%x\n", y);
    return 0;
}
```

This program prints the rotated value of **v** in hexadecimal. Which of the following MIPS code correctly implements the same operation? Assume that the 32-bit unsigned value **v** is stored in register **$t0** and the result **y** should be stored in register **$t1** at the end of the execution.

A.
```
srl $t1, $t0, 4
sll $t2, $t1, 28
or  $t1, $t1, $t2
```

B.
```
srl $t1, $t0, 4
sll $t2, $t0, 28
or  $t1, $t1, $t2
```

C.
```
sll $t1, $t0, 28
srl $t2, $t0, 4
or  $t2, $t1, $t2
```

D.
```
sll $t1, $t0, 28
srl $t2, $t1, 4
or  $t1, $t1, $t2
```

E. None of the above.

11. Which pseudo-instruction corresponds to the following MIPS code?

```
slt $t3, $t1, $t2
beq $t3, $0, L
```

A. `blt $t1, $t2, L` # branch if less than

B. `ble $t1, $t2, L` # branch if less than or equal to

C. `bgt $t1, $t2, L` # branch if greater than

D. `bge $t1, $t2, L` # branch if greater than or equal to

E. None of the above.

12. Consider the following C program:

```
#include <stdio.h>
struct Data {
    int value;
    char label;
};
int main() {
    struct Data arr[3] = {{10,'A'},{20,'B'},{30,'C'}};
    struct Data *p = arr;
    p++;
    p->value += 5;
    printf("%d %c\n", arr[1].value, arr[1].label);
    return 0;
}
```

Which of the following statements best describes the behaviour of this program?

A. The program prints **`10 A`**.
B. The program prints **`20 B`**.
C. The program prints **`25 B`**.
D. The program prints **`30 C`**.
E. The program results in a runtime error.

13. A custom 16-bit Instruction Set Architecture (ISA) defines two instruction types:

`Type A: [opcode: 5 | rs: 3 | rt: 3 | rd: 3 | sha: 2]`
`Type B: [opcode: 6 | rs: 3 | rt: 3 | immediate: 4]`

The ISA must support both instruction types, and the designer wants to modify the encoding to maximize or minimize the number of possible opcodes using an expanding opcode scheme. If the encoding space is to be maximised, which of the following correctly describes the possible minimum and maximum total number of opcode values?

A. Maximum= 64 and Minimum = 32
B. Maximum= 64 and Minimum = 31
C. Maximum= 63 and Minimum = 32
D. Maximum= 63 and Minimum = 31
E. None of the above.

14. Study the MIPS code fragment below.

```
        andi $t0, $s0, 3
        beq  $t0, $0, skip1
        instA
        j    skip2
skip1:  instB
skip2:
```

Suppose variable *a* is mapped to $s0. Which of the following C code corresponds to the above MIPS code?

A.
```
if (a%4 == 3) instA;
else          instB;
```

B.
```
if (a%4 == 3) instB;
else          instA;
```

C.
```
if (a%4 == 0) instA;
else          instB;
```

D.
```
if (a%4 == 0) instB;
else          instA;
```

E. None of the above.

15. How many lines of output does the following C program produce?

```
#include <stdio.h>

int main() {
   float a = 21.0, b = 1.0, sum = 0.0;

   while (b/a < 0.5) {
      b *= 3;
      sum += (b/a);
      printf("%f\n", sum);
   }
   return 0;
}
```

A. 3
B. 4
C. 5
D. The program runs into an infinite loop.
E. None of the above.

Q16 and 17 are based on the following C program:

```
#include <stdio.h>

void foo(int *ptr) {
   *ptr = 10;
   ptr += 1;
}

int main() {
   int arr[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
   int *ptr = arr;
   foo(ptr);
   printf("%d %d\n", *ptr, arr[0]);
   return 0;
}
```

Assume that the base address of **`arr`** is 0x00001000 and each integer occupies 4 bytes in memory.

16. What will be the output of the above program?

A. 1 1
B. 1 10
C. 10 1
D. 10 10
E. None of the above.

17. What will be the value of **`ptr`** in the **main** function after calling the **foo** function?

A. 0x00001000
B. 0x00001004
C. 0x00001008
D. 0x0000100C
E. None of the above.

Q18 – 20 are based on the following MIPS code fragment on a 10-element integer array *A* whose base address is stored in $a0.

```
        addi $t0, $0, 10
        addi $t1, $a0, 40
label:  addi $t1, $t1, -4
        beq  $t1, $a0, exit
        sw   $t0, 0($t1)
        addi $t0, $t0, -1
        j    label
exit:
```

18. What is the final value of *A*[0]?

    A. 0
    B. 1
    C. 10
    D. Insufficient information to determine the final value of *A*[0].
    E. None of the above.

19. What is the sum of *A*[5] and *A*[6] after the execution of the code?

    A. 11
    B. 13
    C. 15
    D. 17
    E. None of the above.

20. How many instructions are executed in total?

    A. 45
    B. 47
    C. 49
    D. 52
    E. None of the above.

Section-B [20 marks]: Fill in the answers within the boxes in the Answer Sheet. Anything written outside the boxes will not be graded.

21. [Total: 4 marks]
Given the following MIPS code fragment on a 10-element integer array *A* whose base address is stored in $a0.

```
        addi $t0, $0, 10
        addi $t1, $a0, 40
label:  addi $t1, $t1, -4
        beq  $t1, $a0, exit
        sw   $t0, 0($t1)
        addi $t0, $t0, -1
        j    label
exit:
```

Write the encoding of the following instructions in hexadecimal on the Answer Sheet. The prefix 0x has been added for you so you don't need to write that. Make sure you write ALL the 8 hexadecimal digits, with A to F in capital letters.

(a) `addi $t1 $a0, 40` [2]

(b) `j label`, assuming that the instruction `j label` is at address **0x4020 883C**. [2]

22. [Total: 4 marks]
Study the MIPS code below. Write your answers in hexadecimal format, filling in ALL the 8 hexadecimal digits, E.g., 0x00123ABC

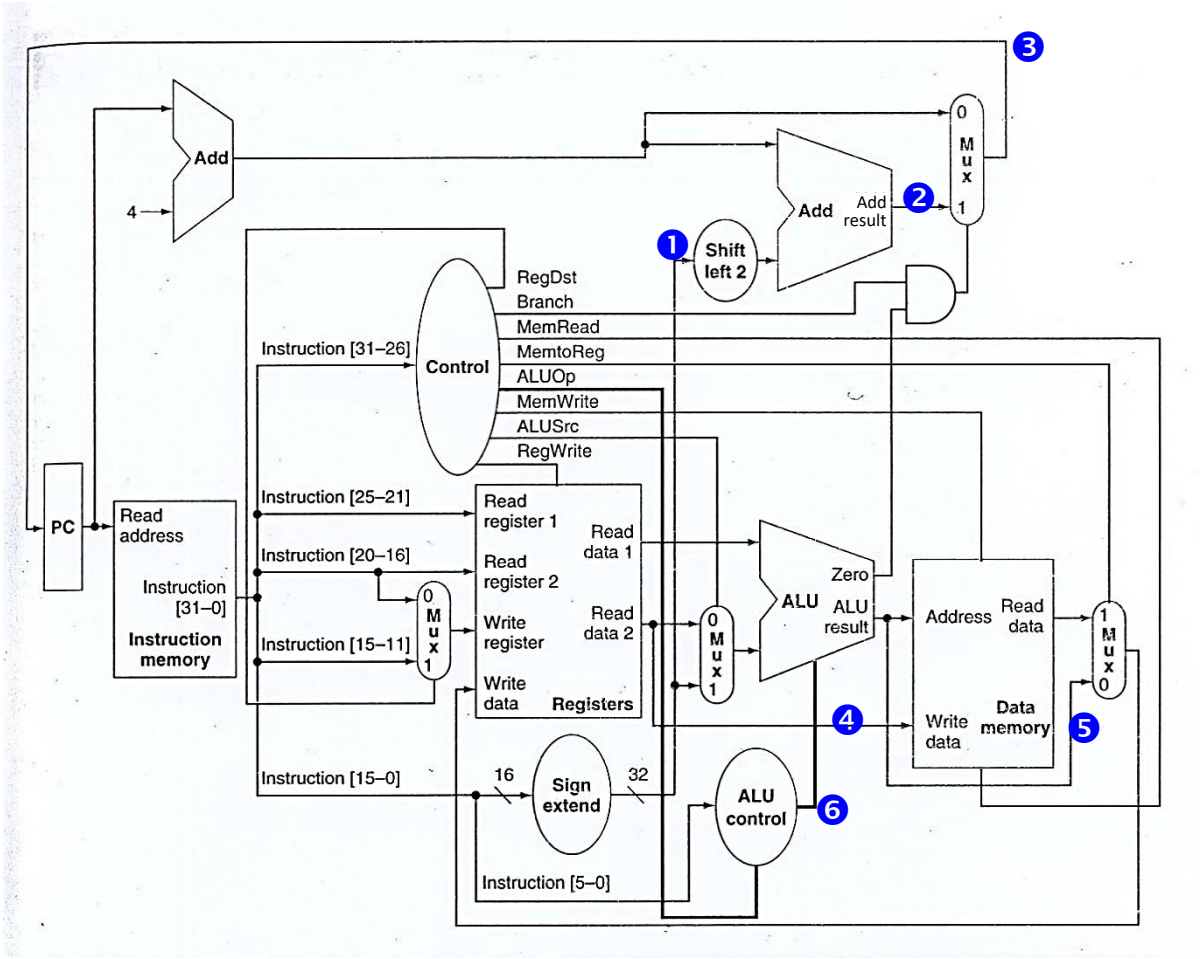
```
addi $t0, $zero, -7   #instruction 1
addi $t1, $zero, 12   #instruction 2
xor  $t0, $t0, $t1    #instruction 3
xor  $t1, $t0, $t1    #instruction 4
xor  $t0, $t0, $t1    #instruction 5
```

(a) What are the values of **$t0** and **$t1** immediately after executing instruction 2? [2]

(b) What are the values of **$t0** and **$t1** immediately after executing instruction 5? [2]

23. [Total: 6 marks]
Consider the MIPS datapath with the hexadecimal values of some registers shown below.

```
R0   [r0] = 0x00000000   |   R10  [t2] = 0x10010028
R1   [at] = 0x10010000   |   R11  [t3] = 0x00000000
R2   [v0] = 0x0000000A   |   R12  [t4] = 0x00000001
R3   [v1] = 0x00000000   |   R13  [t5] = 0x00000000
R4   [a0] = 0x00000005   |   R14  [t6] = 0x00000000
R5   [a1] = 0x10010004   |   R15  [t7] = 0x00000000
R6   [a2] = 0x10018018   |   R16  [s0] = 0x00000000
R7   [a3] = 0x00000000   |   R17  [s1] = 0x00008888
R8   [t0] = 0x00000004   |   R18  [s2] = 0x000003E7
R9   [t1] = 0x10010014   |   R19  [s3] = 0x00000001
```



The instruction being executed is **`sw $s1, -24($t1)`** and this instruction is at address 0x00400050.

Fill in the values of ❶ to ❻ on the Answer Sheet. ❶ is output of Sign extend, ❷ is "Add result", ❸ is output of the multiplexer at the top to the PC, ❹ is "Read data 2", ❺ is "ALU result", and ❻ is output of ALU control.

❶ to ❺ are in hexadecimal and ❻ is in binary. The prefixes 0x and 0b have been added for you on the Answer Sheet, so you do not need to write them. Fill in ALL the 8 hexadecimal digits for ❶ to ❺, with A – F in capital letters, and ALL the 4 bits for ❻.

24. [Total: 6 marks]
Consider the following MIPS code and the memory layout starting at address **0x10010000**. Assume that register **$s0** is initialized to **0x10010000**. The following MIPS instructions are executed:

MIPS Code	Memory	
	Address	**Data (Hexadecimal)**
	0x10010000	FF
`lw $t0, 0($s0)`	0x10010001	12
`lb $t1, 2($s0)`	0x10010002	34
`lb $t2, 4($s0)`	0x10010003	56
`lb $t3, 7($s0)`	0x10010004	78
`lb $t4, 8($s0)`	0x10010005	9A
	0x10010006	BC
	0x10010007	DE
	0x10010008	F0

Note: When the **lb** instruction loads a byte to a MIPS register, it sign-extends the value to 32 bits. What are the values stored in registers **$t0**, **$t1**, **$t2**, **$t3** and **$t4** after executing the above MIPS code? Write your answers in hexadecimal format, filling in ALL the 8 hexadecimal digits, with A to F in capital letters, E.g., 0x00123ABC.

MIPS Reference Data Card ("Green Card") 1. Pull along perforation to separate card 2. Fold bottom side (columns 3 and 4) together

MIPS Reference Data

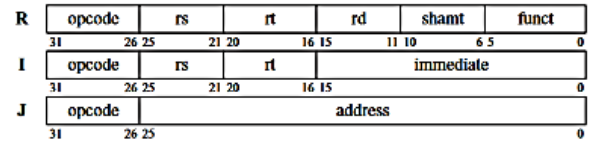
①

CORE INSTRUCTION SET

NAME, MNEMONIC		FOR-MAT	OPERATION (in Verilog)		OPCODE / FUNCT (Hex)
Add	add	R	R[rd] = R[rs] + R[rt]	(1)	0 / 20_{hex}
Add Immediate	addi	I	R[rt] = R[rs] + SignExtImm	(1,2)	8_{hex}
Add Imm. Unsigned	addiu	I	R[rt] = R[rs] + SignExtImm	(2)	9_{hex}
Add Unsigned	addu	R	R[rd] = R[rs] + R[rt]		0 / 21_{hex}
And	and	R	R[rd] = R[rs] & R[rt]		0 / 24_{hex}
And Immediate	andi	I	R[rt] = R[rs] & ZeroExtImm	(3)	c_{hex}
Branch On Equal	beq	I	if(R[rs]==R[rt]) PC=PC+4+BranchAddr	(4)	4_{hex}
Branch On Not Equal	bne	I	if(R[rs]!=R[rt]) PC=PC+4+BranchAddr	(4)	5_{hex}
Jump	j	J	PC=JumpAddr	(5)	2_{hex}
Jump And Link	jal	J	R[31]=PC+8;PC=JumpAddr	(5)	3_{hex}
Jump Register	jr	R	PC=R[rs]		0 / 08_{hex}
Load Byte Unsigned	lbu	I	R[rt]={24'b0,M[R[rs]+SignExtImm](7:0)}	(2)	24_{hex}
Load Halfword Unsigned	lhu	I	R[rt]={16'b0,M[R[rs]+SignExtImm](15:0)}	(2)	25_{hex}
Load Linked	ll	I	R[rt] = M[R[rs]+SignExtImm]	(2,7)	30_{hex}
Load Upper Imm.	lui	I	R[rt] = {imm, 16'b0}		f_{hex}
Load Word	lw	I	R[rt] = M[R[rs]+SignExtImm]	(2)	23_{hex}
Nor	nor	R	R[rd] = ~ (R[rs] | R[rt])		0 / 27_{hex}
Or	or	R	R[rd] = R[rs] | R[rt]		0 / 25_{hex}
Or Immediate	ori	I	R[rt] = R[rs] | ZeroExtImm	(3)	d_{hex}
Set Less Than	slt	R	R[rd] = (R[rs] < R[rt]) ? 1 : 0		0 / $2a_{hex}$
Set Less Than Imm.	slti	I	R[rt] = (R[rs] < SignExtImm)? 1 : 0	(2)	a_{hex}
Set Less Than Imm. Unsigned	sltiu	I	R[rt] = (R[rs] < SignExtImm) ? 1 : 0	(2,6)	b_{hex}
Set Less Than Unsig.	sltu	R	R[rd] = (R[rs] < R[rt]) ? 1 : 0	(6)	0 / $2b_{hex}$
Shift Left Logical	sll	R	R[rd] = R[rt] << shamt		0 / 00_{hex}
Shift Right Logical	srl	R	R[rd] = R[rt] >> shamt		0 / 02_{hex}
Store Byte	sb	I	M[R[rs]+SignExtImm](7:0) = R[rt](7:0)	(2)	28_{hex}
Store Conditional	sc	I	M[R[rs]+SignExtImm] = R[rt]; R[rt] = (*atomic*) ? 1 : 0	(2,7)	38_{hex}
Store Halfword	sh	I	M[R[rs]+SignExtImm](15:0) = R[rt](15:0)	(2)	29_{hex}
Store Word	sw	I	M[R[rs]+SignExtImm] = R[rt]	(2)	$2b_{hex}$
Subtract	sub	R	R[rd] = R[rs] - R[rt]	(1)	0 / 22_{hex}
Subtract Unsigned	subu	R	R[rd] = R[rs] - R[rt]		0 / 23_{hex}

(1) May cause overflow exception
(2) SignExtImm = { 16{immediate[15]}, immediate }
(3) ZeroExtImm = { 16{1'b0}, immediate }
(4) BranchAddr = { 14{immediate[15]}, immediate, 2'b0 }
(5) JumpAddr = { PC+4[31:28], address, 2'b0 }
(6) Operands considered unsigned numbers (vs. 2's comp.)
(7) Atomic test&set pair; R[rt] = 1 if pair atomic, 0 if not atomic

BASIC INSTRUCTION FORMATS

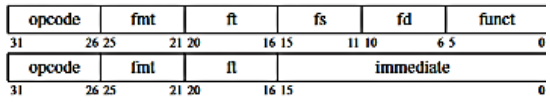


ARITHMETIC CORE INSTRUCTION SET

②

NAME, MNEMONIC		FOR-MAT	OPERATION		OPCODE / FMT /FT / FUNCT (Hex)
Branch On FP True	bc1t	FI	if(FPcond)PC=PC+4+BranchAddr	(4)	11/8/1/--
Branch On FP False	bc1f	FI	if(!FPcond)PC=PC+4+BranchAddr	(4)	11/8/0/--
Divide	div	R	Lo=R[rs]/R[rt]; Hi=R[rs]%R[rt]		0/--/--/1a
Divide Unsigned	divu	R	Lo=R[rs]/R[rt]; Hi=R[rs]%R[rt]	(6)	0/--/--/1b
FP Add Single	add.s	FR	F[fd]= F[fs] + F[ft]		11/10/--/0
FP Add Double	add.d	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} + {F[ft],F[ft+1]}		11/11/--/0
FP Compare Single	c.x.s*	FR	FPcond = (F[fs] *op* F[ft]) ? 1 : 0		11/10/--/*y*
FP Compare Double	c.x.d*	FR	FPcond = ({F[fs],F[fs+1]} *op* {F[ft],F[ft+1]}) ? 1 : 0		11/11/--/*y*
* (x is eq, lt, or le) (*op* is ==, <, or <=) (*y* is 32, 3c, or 3e)					
FP Divide Single	div.s	FR	F[fd] = F[fs] / F[ft]		11/10/--/3
FP Divide Double	div.d	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} / {F[ft],F[ft+1]}		11/11/--/3
FP Multiply Single	mul.s	FR	F[fd] = F[fs] * F[ft]		11/10/--/2
FP Multiply Double	mul.d	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} * {F[ft],F[ft+1]}		11/11/--/2
FP Subtract Single	sub.s	FR	F[fd]=F[fs] - F[ft]		11/10/--/1
FP Subtract Double	sub.d	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} - {F[ft],F[ft+1]}		11/11/--/1
Load FP Single	lwc1	I	F[rt]=M[R[rs]+SignExtImm]	(2)	31/--/--/--
Load FP Double	ldc1	I	F[rt]=M[R[rs]+SignExtImm]; F[rt+1]=M[R[rs]+SignExtImm+4]	(2)	35/--/--/--
Move From Hi	mfhi	R	R[rd] = Hi		0/--/--/10
Move From Lo	mflo	R	R[rd] = Lo		0/--/--/12
Move From Control	mfc0	R	R[rd] = CR[rs]		10/0/--/0
Multiply	mult	R	{Hi,Lo} = R[rs] * R[rt]		0/--/--/18
Multiply Unsigned	multu	R	{Hi,Lo} = R[rs] * R[rt]	(6)	0/--/--/19
Shift Right Arith.	sra	R	R[rd] = R[rt] >>> shamt		0/--/--/3
Store FP Single	swc1	I	M[R[rs]+SignExtImm] = F[rt]	(2)	39/--/--/--
Store FP Double	sdc1	I	M[R[rs]+SignExtImm] = F[rt]; M[R[rs]+SignExtImm+4] = F[rt+1]	(2)	3d/--/--/--

FLOATING-POINT INSTRUCTION FORMATS



PSEUDOINSTRUCTION SET

NAME	MNEMONIC	OPERATION
Branch Less Than	blt	if(R[rs]<R[rt]) PC = Label
Branch Greater Than	bgt	if(R[rs]>R[rt]) PC = Label
Branch Less Than or Equal	ble	if(R[rs]<=R[rt]) PC = Label
Branch Greater Than or Equal	bge	if(R[rs]>=R[rt]) PC = Label
Load Immediate	li	R[rd] = immediate
Move	move	R[rd] = R[rs]

REGISTER NAME, NUMBER, USE, CALL CONVENTION

NAME	NUMBER	USE	PRESERVED ACROSS A CALL?
$zero	0	The Constant Value 0	N.A.
$at	1	Assembler Temporary	No
$v0-$v1	2-3	Values for Function Results and Expression Evaluation	No
$a0-$a3	4-7	Arguments	No
$t0-$t7	8-15	Temporaries	No
$s0-$s7	16-23	Saved Temporaries	Yes
$t8-$t9	24-25	Temporaries	No
$k0-$k1	26-27	Reserved for OS Kernel	No
$gp	28	Global Pointer	Yes
$sp	29	Stack Pointer	Yes
$fp	30	Frame Pointer	Yes
$ra	31	Return Address	No