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. <u>You are to write your answers on the single A4 Answer Sheet provided</u>. 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 <u>correctly</u> with a pencil. Please do this at the start of the assessment and double check that you have shaded it correctly.
- 5. All questions <u>must</u> be answered in the boxes provided on the Answer Sheet. <u>Any</u> 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 autograding software may detect multiple answers.

- 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' (65₁₀) accidentally transferred as 'P' (80₁₀)
 - B. The character '9' (57₁₀) accidentally transferred as 'w' (119₁₀)
 - C. The character 'L' (76₁₀) accidentally transferred as 'z' (122₁₀)
 - D. The character '+' (43₁₀) accidentally transferred as '}' (125₁₀)
 - E. None of the above.
- 3. Given a number system on n bits, where $n \ge 8$, 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. 2ⁿ⁻¹ + 2
 - B. 2ⁿ⁻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) 0x4FFFFFC
 - (iii) 0x5000000
 - (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 \mathbf{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;
}</pre>
```

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, $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) else	<pre>instA; B instB;</pre>	}.	if els	(a%4 e	==	3)	instB; instA;
C. :	if (a%4 == 0) else	<pre>instA; D instB;</pre>).	if els	(a%4 e	==	0)	instB; 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;
}</pre>
```

- 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. 11
- B. 110
- C. 101
- D. 1010
- 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 <u>within the boxes</u> 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, exi	Lt
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 <u>ALL the</u> <u>8 hexadecimal digits, with A to F in capital letters</u>.

(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	Ι	R10 [t2] = 0x10010028
R1	[at] = 0x10010000	Ι	R11 [t3] = 0×00000000
R2	[v0] = 0x000000A	Ι	R12 $[t4] = 0 \times 00000001$
R3	[v1] = 0x00000000	Ι	R13 [t5] = 0×00000000
R4	[a0] = 0x00000005	Ι	R14 [t6] = 0×00000000
R5	[a1] = 0x10010004	Ι	R15 $[t7] = 0x00000000$
R6	[a2] = 0x10010018	Ι	$R16 [s0] = 0 \times 00000000$
R7	[a3] = 0x00000000	Ι	R17 [s1] = 0x00008888
R8	[t0] = 0x0000004	Ι	R18 $[s2] = 0x000003E7$
R9	[t1] = 0x10010014	Ι	R19 [s3] = 0x0000001



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

Fill in the values of 1 to 6 on the Answer Sheet. 1 is output of Sign extend, 2 is "Add result",
is output of the multiplexer at the top to the PC, 4 is "Read data 2", 5 is "ALU result", and
is output of ALU control.

1 to **S** are in hexadecimal and **G** 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 **1** to **S**, with A - F in capital letters, and ALL the 4 bits for **G**.

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:

MIRS Code		Memory			
	WIPS Code	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	FO		

Note: When the **Ib** 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.

he	1
Bet	I
5	ì
4	÷
anc	-
s 3	1
E	1
팅	I
ွိ	I
side	I
Ë	T
otto	Т
qр	Ì
Fol	i
ci.	÷
g	-
cai	-
ate	
par	I
o se	I
n tç	I
atio	Т
for	Т
per	Т
90	Ì
alo	i
Į,	÷
1.1	÷
_	-
÷	-
B	!
8	!
Ę.	I
ĩ	I
p	I
Ű	I
ata	L
e D	L
Suc	T
fer	T
Re	i
S	i
É.	;
-	•

н

			0	6		ARITHMETIC CORE INSTRUCTION SET
INI I L 9	Ref	fer	ence Data	Į.		FOR-
		-				NAME, MNEMONIC MAT OPERATION Branch On FP True boat FL if(FPcond)PC=PC+4+Branch A
LORE INSTRUCTION	UN SE	I FOR	_		/FUNCT	Branch On FP False beir FI if([FPcond)PC=PC+4+BranchA
NAME, MNEMO	NIC	MAT	OPERATION (in Verilog)		(Ilex)	Divide div R Lo=R[rs]/R[rt]; Ili=R[rs]%R[rt]
Add	add	R	R[rd] = R[rs] + R[rt]	(1)	0/20 _{hex}	Divide Unsigned divu R Lo=R[rs]/R[rt]; Hi=R[rs]%R[rt]
dd Immediate	add1	I	R[rt] = R[rs] + SignExtImm	(1,2)	8 _{bex}	FP Add Single add.s FK $F[td] = F[ts] + F[tt]$ FP Add (FIfd) FIfd+11) - (FIfd) FIfd+1
dd Imm. Unsigned	addiu	I	R[rt] = R[rs] + SignExtImm	(2)	9 _{bex}	Double add.d FR $\{r[ta], r[ta+1]\} = \{r[ta], r[ta+1]\}$
dd Unsigned	addu	R	R[rd] = R[rs] + R[rt]		0/21 _{her}	FP Compare Single cx.s* FR FPcond = (F[fs] op F[fl])?1:0
and	and	R	Rirdi = Rirsi & Rirti		0/24 _{bex}	FP Compare $c.x.d^{\bullet}$ FR FP cond = ({F[fs], F[fs+1]}) op
nd Immediate	and1	I	Rirti = Rirsi & ZeroExtimm	(3)	Chex	Double $\{r[n], r[n+1]\}$? 1 * (x is eq. 1t. or 1e) (op is = < or <=) (y is 32, 3c, or 3e)
tranch On Equal	beq	I	if(R[rs]=R[rt]) PC=PC+4+BranchAddr	(4)	4 _{bex}	FP Divide Single atv.s FR F[fd] = F[fs] / F[ft] FP Divide trg {F[fd],F[fd+1]} = {F[fs],F[fs+1]}
ranch On Not Equal	bne	I	if(R[rs]!=R[rt]) PC=PC+4+BranchAddr	(4)	5 _{bex}	Double (F[ft],F[ft+1 FP Multiply Single mu1.s FR F[fd] = F[fs] * F[ft]
ump	1	J	PC=JumpAddr	(5)	2 _{bex}	FP Multiply $mul.d FR$ {F[fd],F[fd+1]} = {F[fs],F[fs+1]}
ump And Link	jal	J	R[31]=PC+8;PC=JumpAddr	ത	3ber	FP Subtract Single sub. s FP FIfdl=FIfd1 - FIfd1
ump Register	1r	R	PC=R[rs]		0 / 08her	FP Subtract protect FFfdLFffd+11} = {FffsLFffs+1
and Data Harden at			R[rt]={24*b0,M[R[rs]		24	Double sub.d FR (Fighter in (Fighter)
oad Byte Onsigned	100	I	+SignExtImm](7:0)}	(2)	24hex	Load FP Single 1wc1 I F[rt]=M[R[rs]+SignExtImm]
.oad Halfword	1hu	I	R[rt]={16`b0,M[R[rs]		25her	Load FP Idc1 [F[rt]=M[R[rs]+SignExtImm]; Double Idc1 [F[rt+1]=M[R[rs]+SignExtImm]
Unsigned			+SignExtImm](15:0)}	(2)	20	Move From Hi mth1 R R[rd] = Ili
.oad Linked	11	1	R[rt] = M[R[rs]+SignExtImm]	(2,7)	30hex	Move From Lo mtio R R[rd] = Lo
.oad Upper Imm.	1u1	I	$R[rt] = \{imm, 16b0\}$		thex	Move From Control mfc0 R $R[rd] = CR[rs]$
oad Word	1W	I	R[rt] = M[R[rs]+SignExtImm]	(2)	23hex	Multiply mult $R \{Hi, I.o\} = R[rs] * R[rt]$
lor	nor	R	R[rd] = ~ (R[rs] R[rt])		0/27 _{hex}	Shift Right Arith. sra R $R[rd] = R[rd] >> shamt$
Dr	or	R	R[rd] = R[rs] R[rt]		0/25 _{hex}	Store FP Single swc1 I M[R[rs]+SignExtImm] = F[rt]
or Immediate	ori	I	R[rt] = R[rs] ZeroExtImm	(3)	dbex	Store FP M[R[rs]+SignExtImm] = F[rt];
et Less Than	slt	R	R[rd] = (R[rs] < R[rt]) ? 1 : 0		0/2a _{bex}	Double M[R[rs]+SignExtImm+4] = F[rt
et Less Than Imm.	slt1	I	R[rt] = (R[rs] < SignExtImm)? 1	: 0 (2)	ahex	FLOATING-POINT INSTRUCTION FORMATS
et Less Than Imm. Unsigned	sltiu	I	R[rt] = (R[rs] < SignExtImm) ? 1 : 0	(2,6)	b _{hex}	FR opcode fmt ft fs fd 31 26 25 21 20 16 15 11 10
et Less Than Unsig.	sltu	R	R[rd] = (R[rs] < R[rt]) ? 1 : 0	(6)	0 / 2b _{hex}	F1 opcode fint ft imm
hift Left Logical	s11	R	R[rd] = R[rt] << shamt		0 / 00 _{hex}	31 26 25 21 20 16 15
hift Right Logical	srl	R	R[rd] = R[rt] >> shamt		0 / 02 _{hex}	PSEUDOINSTRUCTION SET
tore Byte	sb	I	M[R[rs]+SignExtImm](7:0) =		20	NAME MNEMONIC OPER
			R[rt](7:0)	(2)	Zohex	Branch Less Than bit. if(R[rs] <r[rt]) p<="" td=""></r[rt])>
tore Conditional	sc	I	R[rt](7:0) $M[R[rs]+SignExtImm] = R[rt];$ $R[rt] = (atomic) ? 1:0$	(2) (2,7)	38 _{hex}	Branch Less Than DIt. if(R[rs] <r[rt]) p<="" th=""> Branch Greater Than bgt if(R[rs]<r[rt]) p<="" td=""> Branch Less Than or Equal ble if(R[rs]<=R[rt])</r[rt])></r[rt])>
lore Conditional tore Halfword	sc sh	I I	R[rt](7:0) $M[R[rs]+SignExtImm] = R[rt];$ $R[rt] = (atomtc)? 1:0$ $M[R[rs]+SignExtImm](15:0) =$ $R[rt](15:0)$	(2) (2,7)	28 _{bex} 38 _{bex} 29 _{bex}	Branch Less Than Dit. $if(R[rs] < R[rt]) P$ Branch Greater Than bgt $if(R[rs] < R[rt]) P$ Branch Less Than or Equal ble $if(R[rs] <= R[rt])$ Branch Greater Than or Equal bge $if(R[rs] >= R[rt])$ Branch Greater Than or Equal bge $if(R[rs] >= R[rt])$ Load Immediate 11 $R[rd] = immediate$
ore Conditional ore Halfword	sc sh	I I I	R[r1](7:0) M[R[rs]+SignExtImm] = R[r1]; R[r1] = (atomic)? 1:0 M[R[rs]+SignExtImm](15:0) = R[r1](15:0) M[R[rs]+SignExtImm] = R[r1]	(2) (2,7) (2) (2)	28 _{bex} 38 _{bex} 29 _{bex}	Branch Less Than b1t. if(R[rs] <r[rt]) p<="" td=""> Branch Greater Than bgt if(R[rs]<r[rt]) p<="" td=""> Branch Less Than or Equal b1e if(R[rs]<=R[rt])</r[rt])></r[rt])>
tore Conditional tore Halfword tore Word ubtract	sc sh sw	I I I P	R[ri](7:0) M[R[rs]+SignExtImm] = R[rt]; R[rt] = (atomic) ? 1 : 0 M[R[rs]+SignExtImm](15:0) = R[rt](15:0) M[R[rs]+SignExtImm] = R[rt] P[rt] = P[rs] - P[rt]	(2) (2,7) (2) (2) (1)	28 _{bex} 38 _{bex} 29 _{bex} 2b _{bex}	Branch Less Than b1t. if(R[rs] <r[rt]) p<="" td=""> Branch Greater Than bgt if(R[rs]<r[rt]) p<="" td=""> Branch Less Than or Equal b1e if(R[rs]<=R[rt])</r[rt])></r[rt])>
tore Conditional tore Halfword tore Word ubtract ubtract	sc sh sw sub	I I I R P	$R[r](7:0) \\ M[R[rs]+SignExtImm] = R[rt]; \\ R[rt] = (atomic) ? 1 : 0 \\ M[R[rs]+SignExtImm](15:0) = \\ R[rt](15:0) \\ M[R[rs]+SignExtImm] = R[rt] \\ R[rd] = R[rs] - R[rt] \\ P[rd] = P[rs] - P[rt] $	(2) (2,7) (2) (2) (1)	28 _{bex} 38 _{bcx} 29 _{bex} 2b _{bex} 0/22 _{bex}	Branch Less Than b1t. if(R[rs] <r[rt]) p<="" td=""> Branch Greater Than bgt if(R[rs]<r[rt]) p<="" td=""> Branch Less Than or Equal b1e if(R[rs]<=R[rt])</r[rt])></r[rt])>
Store Conditional Store Halfword Store Word Subtract Subtract Unsigned	sc sh sw sub subu (1) May	I I R R v cau	$R[r](7:0) \\ R[r]+SignExtImm] = R[r]; \\ R[r] = (atomic) ? 1 : 0 \\ M[R[rs]+SignExtImm](15:0) = \\ R[rt](15:0) \\ M[R[rs]+SignExtImm] = R[rt] \\ R[rd] = R[rs] - R[rt] \\ R[rd] = R[rs] - R[rt] \\ se overflow excention \\ R[rd] = R[rs] - R[rt] \\ R[rd] = R[rt] \\ R[rt] \\ R[rt] \\ R[rt] = R[rt] \\ R$	(2) (2,7) (2) (2) (1)	28 _{bex} 38 _{bex} 29 _{bex} 2b _{bex} 0 / 22 _{bex} 0 / 23 _{bex}	Branch Less Than b1t. if(R[rs] <r[rt]) f<="" td=""> Branch Greater Than bgt if(R[rs]<r[rt]) f<="" td=""> Branch Less Than or Equal b1e if(R[rs]<r[rt]) f<="" td=""> Branch Greater Than or Equal b1e if(R[rs]<=R[rt])</r[rt])></r[rt])></r[rt])>
Hore Conditional Atore Halfword Atore Word Aubtract Aubtract Unsigned	sc sh sw sub subu (1) May (2) Sign	I I R R y cau nExtl	$R[r](7:0) \\ M[R[rs]+SignExtImm] = R[rt]; \\ R[rt] = (atomic) ? 1 : 0 \\ M[R[rs]+SignExtImm](15:0) = \\ R[rt](15:0) \\ M[R[rs]+SignExtImm] = R[rt] \\ R[rd] = R[rs] - R[rt] \\ R[rd] = R[rs] - R[rt] \\ se overflow exception \\ mm = \{ 16\{immediate[15]\}, immination in the set of the se$	(2) (2,7) (2) (2) (1) ediate	20hex 38hex 29hex 2bhex 0 / 22hex 0 / 23hex }	Branch Less Than b1t. if(R[rs] <r[rt]) p<="" td=""> Branch Greater Than bgt if(R[rs]<r[rt]) p<="" td=""> Branch Less Than or Equal b1e if(R[rs]<<=R[rt])</r[rt])></r[rt])>
Store Conditional Store Halfword Store Word Subtract Subtract Unsigned	sc sh sw sub (1) May (2) Sign (3) Zere	I I R R y cau nExtl oExtl	$R[r1](7:0) \\ R[R[rs]+SignExtImm] = R[rt]; \\ R[rt] = (aiomic)? 1:0 \\ M[R[rs]+SignExtImm](15:0) = \\ R[rt](15:0) \\ M[R[rs]+SignExtImm] = R[rt] \\ R[rd] = R[rs] - R[rt] \\ R[rd] = R[rs] - R[rt] \\ R[rd] = R[rs] - R[rt] \\ se overflow exception \\ mm = \{ 16\{1b^{\circ}0\}, immediate \} \}$	(2) (2,7) (2) (1) ediate	20hex 38 _{hex} 29 _{hex} 2b _{hex} 0 / 22 _{hex} 0 / 23 _{hex} }	Branch Less Than b1t. if(R[rs] <r[rt]) p<="" td=""> Branch Greater Than or Equal b1e if(R[rs]<r[rt]) p<="" td=""> Branch Less Than or Equal b1e if(R[rs]<<=R[rt])</r[rt])></r[rt])>
dore Conditional store Halfword store Word subtract subtract Unsigned	sc sh sw sub subu (1) May (2) Sigi (3) Zere (4) Brat	I I R R y cau nExtl oExtl nchA	$R[r1](7:0)$ $M[R[rs]+SignExtImm] = R[rt];$ $R[rt] = (atomic)? 1:0$ $M[R[rs]+SignExtImm](15:0) = R[rt](15:0)$ $M[R[rs]+SignExtImm] = R[rt]$ $R[rd] = R[rs] - R[rt]$ $R[rd] = R[rs] - R[rt]$ $R[rd] = R[rs] - R[rt]$ se overflow exception mm = { 16{immediate[15]}, imminimm = { 16{immediate	(2) (2,7) (2) (1) ediate ediate	20hex 38 _{hex} 29 _{hex} 20 _{hex} 0/22 _{hex} 0/23 _{hex} } 2'b0 }	Branch Less Than b1t. if(R[rs] <r[rt]) p<="" td=""> Branch Greater Than bgt if(R[rs]<r[rt]) p<="" td=""> Branch Less Than or Equal b1e if(R[rs]<<=R[rt])</r[rt])></r[rt])>
dore Conditional dore Halfword dore Word dubtract dubtract Unsigned	sc sh sub subu (1) May (2) Sign (3) Zera (4) Brat (5) Jum (6) Ope	I I R R y cau nExtI oExtI nchA ipAdi trand	$R[r1](7:0) \\ R[rs]+SignExtImm] = R[rt]; \\ R[rt] = (atomic) ? 1 : 0 \\ M[R[rs]+SignExtImm](15:0) = \\ R[rt](15:0) \\ M[R[rs]+SignExtImm] = R[rt] \\ R[rd] = R[rs] - R[rt] \\ R[rd] = R[rs] - R[rt] \\ R[rd] = R[rs] - R[rt] \\ mm = \{ 16\{1mmediate[15]\}, immin \\ imm = \{ 16\{1mmediate[15]\}, immin \\ r = \{ PC+4[31:28], address, 2't \\ s considered unsigned numbers ('v) \\ R[rs] + SignExt[rs] + R[rt] \\ R[rs] + SignExt[rs] + R[rt] \\ R[rd] = R[rs] - R[rt] \\ R[rd] = R[rt] \\ R[rd] = R[rs] - R[rt] \\ R[rd] = R[rt] \\ R[rd] = R[rt] - R[rt] \\ R[rd] = R[rt] \\ R[rt] = R[rt] \\ R[rd] = R[rt] \\ R[rt] = R[rt] \\ R[rt]$	(2) (2,7) (2) (1) ediate ediate, x0 } s. 2's (20hex 38 _{hex} 29 _{hex} 20 _{hex} 0/22 _{hex} 0/23 _{hex} } 2'b0 }	$\begin{tabular}{l l l l l l l l l l l l l l l l l l l $
lore Conditional tore Halfword tore Word ubtract ubtract Unsigned	sc sh sw sub (1) May (2) Sign (3) Zer (3) Zer (4) Brai (5) Jum (6) Opc (7) Ator	I I R R y cau nExtI oExtI nchA upAde rand mic b	$R[r1](7:0) \\ M[R[rs]+SignExtImm] = R[rt]; \\ R[rt] = (atomic) ? 1 : 0 \\ M[R[rs]+SignExtImm](15:0) = \\ R[rt](15:0) \\ M[R[rs]+SignExtImm] = R[rt] \\ R[rd] = R[rs] - R[rt] \\ R[rd] = R[rs] - R[rt] \\ R[rd] = R[rs] - R[rt] \\ mm = \{ 16\{1b^{-}0\}, immediate[15]\}, immediate [15]], immedi$	(2) (2,7) (2) (1) ediate ediate, x0 } s. 2's (ic, 0 if	20hex 38hex 29hex 20hex 0/22hex 0/23hex } 2'b0 } comp.) not atomic	Branch Less Than b1t. if(R[rs] <r[rt]) p<="" td=""> Branch Greater Than bgt if(R[rs]<r[rt]) p<="" td=""> Branch Less Than or Equal b1e if(R[rs]<r[rt]) p<="" td=""> Branch Greater Than or Equal b1e if(R[rs]<=R[rt])</r[rt])></r[rt])></r[rt])>
Aore Conditional Aore Halfword Aore Word Aubtract Aubtract Unsigned	sc sh sub subu (1) May (2) Sigg (3) Zera (4) Brau (5) Jum (6) Opc (7) Ator ON FO	I I R R v cau nExtI oExtI nchA upAdd rand mic to RMA	$R[r1](7:0) \\ M[R[rs]+SignExtImm] = R[rt]; \\ R[rt] = (atomic) ? 1 : 0 \\ M[R[rs]+SignExtImm](15:0) = \\ R[rt](15:0) \\ M[R[rs]+SignExtImm] = R[rt] \\ R[rd] = R[rs] - R[rt] \\ R[rd] = R[rs] - R[rt] \\ R[rd] = R[rs] - R[rt] \\ mm = \{ 16\{limmediate[15]\}, immin \\ If (16\{lib'0\}, immediate \} \\ ddr = \{ 14\{immediate[15]\}, immin \\ r = \{ PC+4[31:28], address, 2't \\ sc considered unsigned numbers (we stakest pair, R[rt] = 1 if pair atom \\ NTS$	(2) (2,7) (2) (2) (1) ediate ediate, x0 } s. 2's c ic, 0 if	20hex 38 _{hex} 29 _{hex} 20 _{hex} 0/22 _{hex} 0/23 _{hex} } 2'b0 } comp.) not atomic	Branch Less Than b1t. if(R[rs] <r[rt]) p<="" td=""> Branch Greater Than bgt ii(R[rs]<r[rt]) p<="" td=""> Branch Less Than or Equal b1e if(R[rs]<=R[rt])</r[rt])></r[rt])>
Avere Conditional Avere Halfword Avere Word Aubtract Aubtract Unsigned ASIC INSTRUCTION R opcode	sc sh sub subu (1) May (2) Sigi (3) Zeri (4) Brai (5) Jum (6) Opc (7) Alor ON FO	I I R R y cau nExtI oExtI nchA upAda: rand mic b RMA	$\begin{array}{ c c c c c c c c c c c c c c c c c c c$	(2) (2,7) (2) (1) ediate ediate, s0 } s. 2's (ic, 0 if t	20hex 38hex 29hex 20hex 0/22hex 0/23hex } 2*b0 } comp.) not atomic funct	Branch Less Than b1t. if(R[rs] <r[rt]) p<="" td=""> Branch Greater Than bgt ii(R[rs]<r[rt]) p<="" td=""> Branch Less Than or Equal b1e if(R[rs]<=R[rt])</r[rt])></r[rt])>
Nore Conditional Nore Halfword Nore Word Subtract Subtract Unsigned ASIC INSTRUCTION R opcode 31 2	sc sh sub subu (1) May (2) Sigi (3) Zerc (4) Brai (5) Jum (6) Opc (7) Ator ON FO	I I R R y cau nExtI oExtI nchA ipAda yrand mic to RMA	$R[T_{1}(7:0) \\ R[T_{1}]+SignExtImm] = R[T_{1}]; \\ R[T_{1}] = (atomic) ? 1 : 0 \\ M[R[T_{3}]+SignExtImm](15:0) = \\ R[T_{1}](15:0) \\ M[R[T_{3}]+SignExtImm] = R[T_{1}] \\ R[T_{1}] = R[T_{3}] - R[T_{1}] \\ R[T_{2}] = R[T_{3}] - R[T_{1}] \\ R[T_{3}] = R[T_{3}] - R[T_{3}] \\ R[T_{3}] = R[T_{3}] \\ R[T_{3}]$	(2) (2,7) (2) (1) ediate ediate, 50 } s. 2's (ic, 0 if t 6 5	20hex 38 _{bex} 29 _{bex} 20 _{bex} 0/22 _{hex} 0/23 _{hex} } 2'b0 } comp.) not atomic funct 0	Branch Less Than b1t. if(R[rs] <r[rt]) p<="" td=""> Branch Greater Than or Equal b1t. if(R[rs]<r[rt]) p<="" td=""> Branch Greater Than or Equal b1t. if(R[rs]<r[rt]) p<="" td=""> Branch Greater Than or Equal b1t. if(R[rs]<r[rt]) p<="" td=""> Branch Greater Than or Equal b1t. if(R[rs]<r[rt]) p<="" td=""> Branch Greater Than or Equal b1t. if(R[rs]<r[rt]) p<="" td=""> Branch Greater Than or Equal b1t. if(R[rs]<r[rt]) p<="" td=""> Branch Greater Than or Equal b1t. if(R[rs]<r[rt]) p<="" td=""> Branch Greater Than or Equal b1t. if(R[rs]<</r[rt])></r[rt])></r[rt])></r[rt])></r[rt])></r[rt])></r[rt])></r[rt])>
Nore Conditional Nore Halfword Nore Word Subtract Subtract Unsigned ASIC INSTRUCTION R opcode 31 2 I opcode	sc sh sw sub subu (1) May (2) Sigi (3) Zen (4) Brai (5) Jum (6) Opc (7) Ator ON FO 5 25 5 75 5 75	I I R R y cau nExtl oExtl nchA upAdu srand: mic b RMA ; 21	$R[T](7:0) \\ R[R[rs]+SignExtImm] = R[rt]; \\ R[rt] = (atomic) ? 1 : 0 \\ M[R[rs]+SignExtImm](15:0) = \\ R[rt](15:0) \\ M[R[rs]+SignExtImm] = R[rt] \\ R[rd] = R[rs] - R[rt] \\ R[rd] = R[rs] - R[rt] \\ R[rd] = R[rs] - R[rt] \\ se overflow exception \\ mm = \{ 16\{1b^{\circ}0\}, immediate [15] \}, immediate \} \\ ddr = \{ 14\{immediate[15] \}, immediate \} \\ ddr = \{ 14\{immediate[15] \}, immediate \} \\ scalared unsigned numbers (v; est& set pair; R[rt] = 1 if pair atom vistors (v; est& set pair; R[rt] = 1 if pair at$	(2) (2,7) (2) (2) (1) ediate ediate, so } s. 2's s ic, 0 if t <u>6 5</u> liate	20hex 38 _{bex} 29 _{bex} 29 _{bex} 0/22 _{hex} 0/23 _{hex} } 2'b0 } comp.) not atomic funct	Branch Less ThanDIt.if(R[rs] <r[rt]) p<="" th="">Branch Greater ThanDgtif(R[rs]<r[rt]) p<="" td="">Branch Greater Than or EqualD1eif(R[rs]<r[rt]) p<="" td="">Branch Greater Than or EqualD1eR[rd] = mmediaMovemovemoveR[rd] = R[rs]REGISTER NAME, NUMBER, USE, CALL CONVENTIONNAMENUMBERUSESzero0The Constant Value 0\$at1Assembler Temporary\$v0-Sv12-3and Expression Evaluation\$a0-Sa34-7Arguments\$t0-St18-15Temporaries\$s0-Ss716-23Saved Temporaries\$t8-St924-25Temporaries\$k0-Sk126-27Reserved for OS Kernet\$gp28Global Pointer\$gp29Stack Pointer</r[rt])></r[rt])></r[rt])></r[rt])></r[rt])></r[rt])></r[rt])></r[rt])></r[rt])></r[rt])></r[rt])>
Store Conditional Store Halfword Store Word Subtract Subtract Unsigned SASIC INSTRUCTION R opcode 31 2 I opcode 31 2	sc sh sw sub subu (1) May (2) Sigi (3) Zen (4) Brai (5) Jum (6) Opc (7) Ator ON FO 5 25 6 25 1 5	I I R R y cau nExtl oExtl nchA upAda yrand mic (R MA ; 21 ; 21	$R[T](7:0) \\ R[R[rs]+SignExtImm] = R[rt]; \\ R[rt] = (atomic) ? 1 : 0 \\ M[R[rs]+SignExtImm](15:0) = \\ R[rt](15:0) \\ M[R[rs]+SignExtImm] = R[rt] \\ R[rd] = R[rs] - R[rt] \\ R[rd] = R[rs] - R[rt] \\ R[rd] = R[rs] - R[rt] \\ mm = \{ 16\{ lb^{0} 0 \}, immediate [15] \}, immediate [16\{ lb^{0} 0 \}, immediate] \\ ddr = \{ 14\{ immediate[15] \}, immediate] \\ ddr = \{ 14\{ immediate[15] \}, immediate] \\ sconsidered unsigned numbers (v; est& set pair; R[rt] = 1 if pair atom \\ VTS \\ \hline rt rd shamediate [20, rt] \\ 20 16 15 11 10 \\ \hline rt & immediate] \\ \hline rt & rt & immedi$	(2) (2,7) (2) (1) ediate ediate, so } s. 2's s ic, 0 if <u>6 5</u> <u>liate</u>	20hex 38 _{bex} 29 _{bex} 29 _{bex} 0/22 _{hex} 0/23 _{hex} } 2'b0 } comp.) not atomic funct	$\begin{tabular}{ c c c c c } \hline Branch Less Than & bit. if(R[rs]$
Store Conditional Store Halfword Store Word Subtract Subtract Unsigned SASIC INSTRUCTION R opcode 31 2 J opcode 31 2 31 2	sc sh sw subu (1) May (2) Sigi (3) Zeri (4) Brat (5) Jum (6) Opc (7) Ato ON FO ON FO CON FO	I I R R y cau nExtl oExtl nchA upAda stand mic b RMA ; 21	$R[T](7:0) \\ R[R[rs]+SignExtImm] = R[rt]; \\ R[rt] = (atomic) ? 1 : 0 \\ M[R[rs]+SignExtImm](15:0) = \\ R[rt](15:0) \\ M[R[rs]+SignExtImm] = R[rt] \\ R[rd] = R[rs] - R[rt] \\ R[rd] = R[rs] - R[rt] \\ se overflow exception \\ mm = \{ 16\{inmediate[15]\}, imm \\ mm = \{ 16\{inmediate[15]\}, imm \\ mm = \{ 16\{inmediate[15]\}, imm \\ dr = \{ 14\{inmediate[15]\}, imm \\ sc onsidered unsigned numbers (v. est&set pair, R[rt] = 1 if pair atom \\ xrs \\ rt rd shame \\ 20 16 15 11 10 \\ \hline rt ct sc onstant \\ address \\ \hline rt ct sc onstant \\ rt sc onstant \\ rt sc onstant \\ rt sc onstant \\ rt ct sc onstant \\ rt sc$	(2) (2,7) (2) (1) ediate ediate, so } s. 2's s ic, 0 if <u>6 5</u> liate	20hex 38 _{bex} 29 _{bex} 0/22 _{hex} 0/23 _{hex} } 2'b0 } comp.) not atomic funct 0 0	$\begin{tabular}{ c c c c c } \hline Branch Less Than & bit. & if(R[rs]$

OPCODE

M[R[rs]+SignExtImm] = F[rt]; M[R[rs]+SignExtImm+4] = F[rt+1]

{F[n],F[n+1]}

{F[ft],F[ft+1]}

{F[ft],F[ft+1]}

{F[ft],F[ft+1]}

F1 if(FPcond)PC=PC+4+BranchAddr (4)

FI if(!FPcond)PC=PC+4+BranchAddr(4)

add.d FR {F[fd],F[fd+1]} = {F[fs],F[fs+1]} +

c.x.d* FR FPcond = ({F[fs],F[fs+1]} op {F[n],F[ft+1]})?1:0 r1e) (op is =, <, or <=) (y is 32, 3c, or 3e)

div.d FR {F[fd],F[fd+1]} = {F[fs],F[fs+1]} /

mul.d FR {F[fd],F[fd+1]} = {F[fs],F[fs+1]} *

sub.d FR $\{F[fd],F[fd+1]\} = \{F[fs],F[fs+1]\}$

F[rt]=M[R[rs]+SignExtImm] F[rt]=M[R[rs]+SignExtImm]; F[rt+1]=M[R[rs]+SignExtImm+4] / FMT /FT

/ FUNCT

(Hex)

11/8/1/--

11/8/0/---

0/--/-/1a

0/--/--/1b

11/10/--/0

11/11/--/0

11/10/--/y

11/11/—/y

11/10/--/3

11/11/--/3

11/10/--/2

11/11/-/2

11/10/-/1

11/11/--/1

35/--/--/--

0/--/--/10

0/--/-/12

10 /0/--/0

0/--/-18

0/-/-/3

(6) 0/-/-/19

(2) 39/--/--/--

3d/--/--/--

funct

0

0

(2) 31/--/--

(2)

(2)

fd

immediate

OPERATION

PRESERVEDACROSS

A CALL?

N.A.

No

No

No

No

Yes

No

No

Yes

Yes

Yes

No

if(R[rs] < R[rt]) PC = Labelif(R[rs] > R[rt]) PC = Label

if(R[rs]<=R[rt]) PC = Label if(R[rs]>=R[rt]) PC = Label

R[rd] = immediate

(6)