Name _____ Student ID _____

# CSc 252: Computer Organization

## Spring 2025

Test 4 - SOLLUTIONS: Mar. 27, 2025

Time: 30 min

**DO NOT OPEN THE EXAM UNTIL INSTRUCTED TO DO SO**

Please read this page and follow the directions before proceeding with the rest of the exam.

- To give all students the same amount of time to do the exam, please DO **NOT** OPEN THIS EXAM UNTIL INSTRUCTED TO DO SO.
- You are not allowed to use any external resources such as cellphones, notes, headphones, watch, neighbors, calculator, etc. If you have not done so yet, turn your cellphone off and place it in your backpack.
- The cellphone cannot be on you during the exam. If your cellphone is in your pocket, it will be considered cheating even if you are not looking at it (same for headphones and watches). We will collect your exam and ask you to leave.
- Place your final answers in the given boxes. You can show your work on any blank spaces.
- We recommend skimming the entire exam before completing any problems.
- Read carefully every question before answering and raise your hand if the question is unclear.
- DO NOT SPEAK TO ANYONE AT YOUR TABLE.

****** Good Luck! ******

### Allowable MIPS Instructions

When writing MIPS assembly, the only instructions that you are allowed to use (so far) are:

- `and, andi, or, ori, nor, nori, xor, xori`
- `add, addi, sub, addu, addiu`
- `beq, bne, j, jal, jr`
- `mult, div, mfhi, mflo`
- `slt, slti, sll, srl, sra`
- `lw, lh, lb, sw, sh, sb`
- `lui, la`
- `syscall`

*Write your name and student ID on all the exam pages for one extra credit point.*

Name _____ Student ID _____

1. (2 points) How many bits are available for the "immediate" field (that is, the constant) in an I-format instruction?

Place your final answers here.

16

Mark your responses by **completely darkening the circle** with the correct answer. e.g., ●

2. (2 points) Which register(s) points to the top of the stack to manage the function call frame?

| | | | | | |
|---|---|---|---|---|---|
| ○ | $at | ○ | $t2 | ○ | fun1 |
| ○ | $v0 | ○ | $t3 | ○ | jal |
| ○ | $v1 | ○ | $t4 | ○ | $k0 |
| ○ | $a0 | ○ | $s0 | ○ | $k1 |
| ○ | $a1 | ○ | $s1 | ○ | $gp |
| ○ | $a2 | ○ | $s2 | ● | $sp |
| ○ | $a3 | ○ | $s3 | ○ | $fp |
| ○ | $a4 | ○ | $s4 | ○ | $ra |
| ○ | $t0 | ○ | $s5 | ○ | pc |
| ○ | $t1 | ○ | $s6 | ○ | hi |

3. (2 points) Which register(s) is(are) used to store the return value of a function?

| | | | | | |
|---|---|---|---|---|---|
| ○ | $at | ○ | $t2 | ○ | fun1 |
| ● | $v0 | ○ | $t3 | ○ | jal |
| ● | $v1 | ○ | $t4 | ○ | $k0 |
| ○ | $a0 | ○ | $s0 | ○ | $k1 |
| ○ | $a1 | ○ | $s1 | ○ | $gp |
| ○ | $a2 | ○ | $s2 | ○ | $sp |
| ○ | $a3 | ○ | $s3 | ○ | $fp |
| ○ | $a4 | ○ | $s4 | ○ | $ra |
| ○ | $t0 | ○ | $s5 | ○ | pc |
| ○ | $t1 | ○ | $s6 | ○ | hi |

4. (2 points) Which register(s) hold(s) the argument(s) when a function is called?

| | | | | | |
|---|---|---|---|---|---|
| ○ | $at | ○ | $t2 | ○ | fun1 |
| ○ | $v0 | ○ | $t3 | ○ | jal |
| ○ | $v1 | ○ | $t4 | ○ | $k0 |
| ● | $a0 | ○ | $s0 | ○ | $k1 |
| ● | $a1 | ○ | $s1 | ○ | $gp |
| ● | $a2 | ○ | $s2 | ○ | $sp |
| ● | $a3 | ○ | $s3 | ○ | $fp |
| ○ | $a4 | ○ | $s4 | ○ | $ra |
| ○ | $t0 | ○ | $s5 | ○ | pc |
| ○ | $t1 | ○ | $s6 | ○ | hi |

Name _____ Student ID _____

5. (2 points) Which register(s) store(s) the return address(es) of a function call?

| | | | | | |
|---|---|---|---|---|---|
| O | $at | O | $t2 | O | fun1 |
| O | $v0 | O | $t3 | O | jal |
| O | $v1 | O | $t4 | O | $k0 |
| O | $a0 | O | $s0 | O | $k1 |
| O | $a1 | O | $s1 | O | $gp |
| O | $a2 | O | $s2 | O | $sp |
| O | $a3 | O | $s3 | O | $fp |
| O | $a4 | O | $s4 | ● | $ra |
| O | $t0 | O | $s5 | O | pc |
| O | $t1 | O | $s6 | O | hi |

6. (2 points) When decoding a machine instruction, which field(s) determines the type of instruction format (that is, R-format, I-format, or J- format)?

| | | | | | |
|---|---|---|---|---|---|
| ● | opcode | O | address | O | shiftVal |
| O | rs | O | baseReg | O | shiftBits |
| O | rt | O | targetReg | O | funcCode |
| O | rd | O | destReg | O | aluFunc |
| O | shamt | O | resultReg | O | immVal |
| O | funct | O | writeReg | O | offset |
| O | immediate | O | shiftAmt | O | constant |

7. (2 points) Select the format type for each machine instruction.

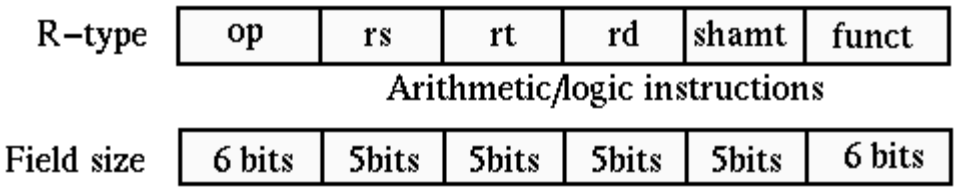
| **Machine instruction in binary** | | | |
|---|---|---|---|
| 0000 1000 0000 0000 0000 0000 0010 0000 | O R-format | O I-format | ● J-format |
| 0000 0001 0010 1010 0100 0000 0010 0000 | ● R-format | O I-format | O J-format |
| 0010 0001 0010 1000 0000 0000 0000 0101 | O R-format | ● I-format | O J-format |
| 0000 0010 0011 0010 1000 0000 0010 0010 | ● R-format | O I-format | O J-format |
| 0000 1000 0000 0000 0000 0001 0000 0000 | O R-format | O I-format | ● J-format |
| 1000 1101 0010 1000 0000 0000 0000 0100 | O R-format | ● I-format | O J-format |
| 0001 0010 0001 0001 0000 0000 0001 0000 | O R-format | ● I-format | O J-format |
| 0000 0000 1010 0110 0010 0000 0010 0100 | ● R-format | O I-format | O J-format |

Name _____ Student ID _____

8. (10 points) Use the tables found in the formula sheet to decode the following instruction. Fill out the corresponding fields in decimal and write the decoded instruction in MIPS.

**0x012A4020**

1. Hexa to binary:
0000 0001 0010 1010 0100 0000 0010 0000
2. Type of instruction?
Opcode = 0000 00 = 0 therefore R-format




3. Rs =01 001 = 9 in decimal
4. Rt = 0 1010 = 10 in decimal
5. Rd = 0100 0 = 8 in decimal
6. Shamt = 000 00 = 0 in decimal
7. Funct = 10 0000 = 32 in decimal

| | Value in Decimal |
|---|---|
| opcode | 0 |
| rs | 9 |
| rt | 10 |
| rd | 8 |
| shamt | 0 |
| funct | 32 |
| immediate | ------- |
| address | -------- |

Place your final MIPS instruction here.

**add $t0, $t1, $t2**

Name _____ Student ID _____

9. (10 points) Use the tables found in the formula sheet to encode the first MIPS instruction (`beq`) from the following code snipped. The other instructions are there just to compute the immediate field of the `beq` instruction. Fill out the corresponding fields in decimal and write the encoded instruction in hexadecimal.

```
   beq $s0, $s1, next
   li $v0, 4
   la $a0, message
   syscall
next:
   li $v0, 10
   syscall
```

1. Beq is an i-format
2. Opcode from the table = 4 = 000100 in binary
3. Rs = $s0 from the table = 16 =10000 in binary
4. Rt = $s1 from the table =17 = 10001 in binary
5. Offset calculation

Using the formula: NEWPC = PC+4 + offset*4
PC = 0 (or any value)
NEWPC = 16 (or any value +16 or PC+16)
Plugin these values in the formula:
16 = 0+4 + offset*4
Offset = (16-4)/4 = 3 = 0000000000000011

6. All the binary together and convert to hexadecimal

00010010000100010000000000000011

| | Value in Decimal |
|---|---|
| opcode | 4 |
| rs | 16 |
| rt | 17 |
| rd | ----- |
| shamt | ----- |
| funct | ----- |
| immediate | 3 |
| address | ----- |

Place the final instruction in hexadecimal here.
One nibble on each box

0x | 1 | 2 | 1 | 1 | 0 | 0 | 0 | 3 |

Name _____ Student ID _____

10. (10 points) Fill in the table below; write each instruction's proper CPU control bits. Refer to the CPU design, which has been included on the formula sheet. Some of the instructions might not be possible, given our standard design. If this is the case, write "impossible" in the row.

- For "do not care" values, you may write **X** or **0**, but 1 will not be accepted.
- For the aluOp, you may use the operation names (such as ADD) or the encodings (such as 2).

| Instruction | ALUsrc | aluOp | bNegate | Branch | Jump | MemWrite | MemRead | MemToReg | RegDst | RegWrite |
|---|---|---|---|---|---|---|---|---|---|---|
| ADD | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| SLT | 0 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| AND | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| J | X | X | X | X | 1 | 0 | 0 | X | X | 0 |
| JR | Not possible | | | | | | | | | |
| BEQ | 0 | 2 | 1 | 1 | 0 | 0 | 0 | X | X | 0 |
| LB | Not possible | | | | | | | | | |
| LW | 1 | 2 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| SRA | Not possible | | | | | | | | | |
| ADDI | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

11. (6 points) Implement the following as an MIPS function. Use comments to show what register is associated with each variable name clearly and which part of the Java code.

```
int fun1(int a, int b) {
    if (a == 0 || b == 0)
        return 0;
    if (a < b)
        return b;
    else
        return a;
}
```

Place your final answer here.

```
# int fun1(int a, int b) {
# a0 = a
# a1 = b
FUN1:
      #prologue
      addiu  $sp, $sp, -24
      sw     $fp, 0($sp)
      sw     $ra, 4($sp)
      addiu  $fp, $sp, 20

    # if (a == 0 || b == 0)
    beq    $a0, $zero, ZERO    # if a == 0, return 0
    beq    $a1, $zero, ZERO    # if b == 0, return 0

    # if (a < b)
    slt    $t0, $a0, $a1              # $t0 = 1 if (a < b)
    bne    $t0, $zero, RETURN_B       # if $t0 == 1, return b

    # else return a;
    add    $v0, $zero, $a0
    j END

RETURN_B:
    add    $v0, $zero, $a1
    j END

ZERO:
    add    $v0, $zero, $zero

END:
      #epilogue
      lw     $ra, 4($sp)
      lw     $fp, 0($sp)
      addiu  $sp, $sp, 24
      jr     $ra
```