Name _____ Student ID _____

CSc 252: Computer Organization

Spring 2025

Test 5: Apr. 10, 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`
- `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. In each part below, list the dependencies between instructions. I have numbered the instructions for convenience. List all the dependencies - even the ones that might be solved because they are far enough apart.
   (Note that some instructions might depend on two others.)
   You may not need to use all these lines; leave extras blank!
   a) (5 points)

```
slt  $s3, $s4, $s5 # instruction 1
sub  $s1, $s2, $s3 # instruction 2
add  $s0, $s1, $s3 # instruction 3
addi $s0, $s3, 123 # instruction 4
```

The __rt_ field in instruction __2____ depends on the instruction ___1____

The __rs_ field in instruction __3____ depends on the instruction ___2____

The __rt_ field in instruction __3____ depends on the instruction ___1____

The __rs_ field in instruction __4____ depends on the instruction ___1____

   b) (5 points)

```
sw   $s0, 0($s1)   # instruction 1
lw   $s0, 0($s2)   # instruction 2
lw   $s3, 0($s0)   # instruction 3
add  $s4, $s0,$s3 # instruction 4
```

The __rs_ field in instruction __3____ depends on the instruction ___2____

The __rs_ field in instruction __4____ depends on the instruction ___2____

The __rt_ field in instruction __4____ depends on the instruction ___3____

Name _____ Student ID _____

2. (3 points) Write a MIPS instruction to multiply the value stored in $s1 by 16, but you cannot use a multiply instruction.

Place your final answers here (a single instruction).

```
sll $s1, $s1, 4
```

3. (3 points) Draw the pipeline diagram for the following instructions on a non-pipelined system.

	Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	add $s1,$s2,$s3	F	D	E	M	W											
2	add $s2,$s1,$s3						F	D	E	M	W						
3	add $s3,$s2,$s1											F	D	E	M	W	
4																	

How many clock cycles does it take to run this program on this system? __15__

4. (6 points) Draw the pipeline diagram for the exact instructions in Question 2 on a pipelined machine that forwards when a data hazard occurs.

	Instruction	Forward?	1	2	3	4	5	6	7	8	9	10
1	add $s1,$s2,$s3	N	F	D	E	M	W					
2	add $s2,$s1,$s3	Y, S1=I1		F	D	E	M	W				
3	add $s3,$s2,$s1	Y,S2=I3,S1=I1			F	D	E	M	W			
4												
5												

How many clock cycles does it take to run this program on this system? __7___

Name _____ Student ID _____

5. (3 points) Fill in the blanks to complete the following prologue and epilogue for a function with seven parameters.

```
# prologue

addiu   $sp, $sp, ___-36____

sw      $fp, ___0__($sp)

sw      $ra, ___4__($sp)

addiu   $fp, $sp, ___32___


# body


# epilogue

lw      $ra, ___4___($sp)

lw      $fp, ___0____($sp)

addiu   $sp, $sp, ___36___

jr      $ra
```

6. (7 points) Implement the following as a MIPS function. Use comments to show what register is associated with each variable name clearly and which part of the original code.

```
int count_matches(int [] data, int dataLen, int needle) {
    int retval = 1;
    for (int i=0; i<dataLen; i++)
        if (data[i] == needle)
            retval++;
    return retval;
}
```

Place your answer on the next page. Note that the prologue and epilogue are given.

Name _____ Student ID _____

```
count_matches:
     # prologue
      addiu  $sp, $sp, -24
      sw     $fp, 0($sp)
      sw     $ra, 4($sp)
      addiu  $fp, $sp, 20


     addi    $t0, $zero,1    # retval

     addi    $t1, $zero,0    # i
LOOP:
     slt     $t2, $t1,$a1
     beq     $t2,$zero, DONE

     sll     $t3, $t1,2
     add     $t3, $a0,$t3
     lw      $t3, 0($t3)

     bne     $t3,$a2, NOT_EQ

     addi    $t0, $t0,1

NOT_EQ:
     addi    $t1, $t1,1
     j       LOOP

DONE:
     add     $v0, $t0,$zero









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

Name _____ Student ID _____

7. (10 points) Implement the following recursive MIPS function. I've provided the prologue and epilogue for you. (Pay special attention to what registers need to be saved.)

```
int foo(int x) {
     return foo(x+1) + foo(x-1);
}
```

Place your answer here.

```
#standard prologue
      addiu  $sp, $sp, -24
      sw     $fp, 0($sp)
      sw     $ra, 4($sp)
      addiu  $fp, $sp, 20

#save sX if the function uses any

#store x, ideally pushed to the stack = +1
      sw     $a0, 8($sp)

# startup sequence:
#save tX if the function uses any
#call foo(x+1)= 1

      addi   $a0, $a0, 1
      jal    foo

#cleanup sequence:
#restore tX if the function uses any

#retrieve x = +1
      sw     $a0, 8($sp)

# startup sequence:
#save tX if the function uses any

#save the value of foo(x+1
      addiu  $sp, $sp, -4
      sw     $v0, 0($sp)

#call foo(x-1) = +1
      addi   $a0, $a0, -1
      jal    foo

#cleanup sequence:
#restore tX if the function uses any
```

Name _____ Student ID _____

```
#computes foo(x+1) + foo(x-1)
      lw     $t0, 0($sp)
      addiu  $sp, $sp, 4

      add    $v0, $v0, $t0

#returns the sum

#restore sX if the function uses any
#standard epilogue

      lw     $ra, 4($sp)
      lw     $fp, 0($sp)
      addiu  $sp, $sp, 24
      jr     $ra
```

8. (8 points) Consider the following function, written in MIPS assembly; complete the function's body in C or Java.

Write your answer as a **single** statement. That is, you may only use **one** semicolon in the entire answer!

JAVA OR C CODE:

```
int whatzit() {
```

//Place your final answer here.

```
return doA(1) + doB(2) + doA(3);

or

return doB(2) + 2 * doA(3);
```

```
}
```

MIPS CODE:

```
.globl whatzit
whatzit:
     addiu $sp, $sp,-24
     sw    $ra, 4($sp)
     sw    $fp, 0($sp)
     addiu $fp, $sp,20
```

Name _____ Student ID _____

```
addi  $a0, $zero, 1
jal doA
addi  $t0, $v0, $zero
addiu $sp, $sp, -4
sw    $t0, 0($sp)
addi  $a0, $zero, 2
jal doB
addi  $t1, $v0, $zero
addiu $sp, $sp, -4
sw    $t1, 0($sp)
addi  $a0, $zero, 3
jal doA
addi  $t2, $v0, $zero
lw    $t1, 0($sp)
lw    $t0, 4($sp)
addiu $sp, $sp, 8
add   $v0, $v0, $t1
add   $v0, $v0, $t2
lw    $fp, 0($sp)
lw    $ra, 4($sp)
addiu $sp, $sp,24
jr    $ra
```