Computer Architecture and Organization

Test 1 2

Time limit: 55 + 5 minutes

15 Points

Jan 29, 2021

Name: Hayvin Bolton

Notes:

- You must not communicate with anybody about the Test questions while you are taking the Test.
- If a question is not clear enough, or if you think you need more information to answer a question, you may make a reasonable assumption. However, you may lose some points based on your assumption.
- I strongly recommend that you take the test in the virtual classroom. This way you will be notified of the possible typos and my hints, if any.
- There are 4 questions on this test.
- Test is open book.
- Type/write/draw clearly, neatly, and concisely in this handout.
- Do not spend too much time on one single question, otherwise you may run out of time.
- Convert this handout into one single .pdf file, and then submit it through Blackboard BEFORE 9:00 am today. Note that the submission link will close right after 9:00 am.
- Check your .pdf file to make sure it is generated properly.
- To submit your Test, please go to **General Resources > Test01** in Blackboard.

1. In this question, let us assume that the initial content of each register is equal to **its address + 6**.

(2 points) Translate the following MIPS assembly instruction into a MIPS machine instruction in binary and hexadecimal. (Let us assume that opcode of sw = **0x26**)

sw $t4 4($t2)

10 0110

Binary	110111	01100	01010	0000 0000 0000 0101

Your opcode is an odd number, while 26 is an even number.

0xDD8A 0004

(2 points) Determine the hexadecimal contents of the memory location(s) affected by the above instruction along with the byte addresses, and type them in the following table. Leave the unused rows blank:

Byte Address	Contents
0x00	0xD
0x00	0xD
0x00	0x8
0x12	0xA
0x	0x
0x	0x
0x	0x

14 00
15 00
16 00
17 12

(1 point) How many numbers (values) will be sign extended for this instruction? Briefly explain:

Only the offset will be sign extended as it needs to be extended to 32 bits

2. (3 points) Write the machine code (in binary and hexadecimal) for the following branch instruction so that it can reach the *farthest* possible instruction (when the bne is successful). Let us assume that the opcode of bne is 0x1C.

Forward bne $s4, $s7, done

Binary	011100	10100	10111	1111 1111 1111 1111

0x7297 FFFF

This offset < 0
we need a positive offset:
0111 1111 1111 1111

2

3. (3 points) Analyze the following assembly code rigorously, and then briefly but clearly explain what it performs, and what the content of $t2 would be when label done is reached. In the meantime, add appropriate comments to the code. Let us assume that pointer $s0 initially points to the base of a byte array that contains ALL the **255** 8-bit **non-zero** values. **The last byte of array is 0.** The array size is around 3000 bytes.

Note

- **Do NOT** explain the details! Your answer should be in this format:
  This code adds 3 consecutive bytes, and places the sum in $t2. The first byte is pointed to by $s0. The content of $t2 will be 125 decimal when the nop is reached.
- In your comments and explanation, avoid using register names (such as $s4) as much as possible.

Instructions			Comments
	lb	$t2, 0($s0)	Load array Load 1st number in max.
	beq	$t2, $zero, done	If at end of array then exit
again:	lb	$t1, 1($s0)	Load array pointer Read next number
	addi	$s0, $s0, 1	Increment array Increment pointer
	beq	$t1, $zero, done	If end of array exit
	slt	$s1, $t2, $t1	Shift array if current array value greater than previous check if max < current byte
	beq	$s1, $zero, again	If Array points to 0 do again if not, continue
	add	$t2, $t1, $zero	Stores current array value in array place holder Otherwise, update max
	beq	$t1, $t1, again	Go again
done:	nop		Break point place holder

Briefly but clearly explain what this program does. What are the inputs and outputs of the program?

signed

The programs goes through the array looking for the largest number in the array and stores it in $t2 which will be the 255 8 bit non zero value.

I assume this is what you mean: 0000 0000 0000 0000 0000 0000 1111 1111

$t2 (in binary) = 1111 1111 This is the unsiged largest, but we need is the signed largest.

0000 0000 0000 0000 0000 0000 0111 1111

4. The "Save Registers" portion of test, a non-leaf procedure, is shown here:

```
test:   sw      $ra,   -4($sp)      # push return address and 3 s-type registers
        sw      $s1,  -8($sp)       #
        sw      $s0,  -12($sp)      #
        sw      $s6,  -16($sp)
```

1

(1 point) Right after these lines, and in the space proved below, write an assembly instruction to update $sp properly:

```
addi $sp, $sp, -16
```

✓

0.5

(1.5 points) In the space provided below, write a piece of assembly code to restore the above registers properly (as many registers as necessary):

```
addi    $sp, $sp, -16
lw      $ra,   -4($sp)      offset = +12
lw      $s1,  -8($sp)       offset = +8
lw      $s0,  -12($sp)      offset = +4
lw      $s6,  -16($sp)
                offset = 0
```

0

Typo?
OK!

✗

0.5

(1.5 points) Back to the first part of this question, where there are 4 sw instructions followed by the "update $sp instruction" that you added. Let us call these code lines version I.

In this section, you will write version II: Rewire version I so that now the instruction that updates $sp comes **first**, while the outcomes of version I and version II are the same, i.e., the two versions can be used interchangeably.

```
addi    $sp, $sp,-16
lw      $ra,   -12($sp)     offset = +12
lw      $s1,  -8($sp)       offset = +8
lw      $s0,  -4($sp)       offset = +4
lw      $s6,  ($sp)
```

✗