CS2100 Computer Organization
2020/21 Semester 2
Mid-term Assessment (Backup)

1. This assessment paper is valid only when instructed to be used by the course coordinators, via the proctors.

2. Answer the questions on a piece of paper. Listen to instructions from the proctors on the ending time of the assessment.

3. When the proctors have told you to stop writing, you have 5 minutes to scan your answers using your mobile phone. You can use software like CamScanner, or simply take a picture. In either case ensure that your writing is legible.

4. Answers that cannot be read will not be marked and you will receive 0 for those questions.

5. When you have finished scanning, email your scans to colin.mod.mails@gmail.com.

6. All restrictions and instructions in the midterm SOP continue to apply.

7. There are FIFTEEN (15) questions on EIGHT (8) printed pages, including this one.

MRQ Questions

Choose ALL options that are correct, and none that are wrong. Each question is worth 2 marks. No partial credits.

1. Given the hexadecimal value 0x3F2C, choose all of the possible interpretations of this value below:

    a. A 16-bit 1's complement negation of -16173
    b. The decimal value 16172.
    c. A 16-bit 2's representation of 16173.
    d. The ASCII characters '?' and ','
    e. The C string "?,"

2. We wish to implement the pseudo-instruction BLE $s1, $s2, target, which branches to "target" if $s1<=$s2. Choose all of the sequences below that implement this behaviour correctly (Note: We are only looking for correct behaviour. May not necessarily be the shortest possible implementation.)

    a. slt $t1, $s1, $s2
       bne $t1, $zero, target

    b. sub $t1, $s1, $s2
       slt $t2, $t1, $zero
       bne $t2, $zero, target

    c. +slt $t1, $s2, $s1
       beq $t1, $zero, target

    d. +sub $t1, $s2, $s1
       slt $t2, $t1, $zero
       beq $t2, $zero, target

    e. slt $t1, $s2, $s1
       bne $t1, $zero, target

3. Aiken has written a program to multiply a number stored in register $s0 by 3 and store the result in register $s1. Sadly he lost the assembly code and only has the following fragments of MIPS machine code in hexadecimal. Help Aiken to find which instructions are from his program. The ordering of the instructions is not important, and we further assume that the register $s2 contains the value 3.

    a. 0x02128818
    b. 0x02308820
    c. 0x02008840
    d. 0x02118824
    e. 0x00108840

4. Pick all of the statements below that are TRUE about the MIPS datapath.

    a. A single cycle implementation is better than multicycle implementation since every MIPS instruction takes exactly the same amount of time to go through the datapath.
    b. A multicycle implementation is better than single-cycle since not all instructions need to pass through every stage of the datapath.
    c. In an N stage datapath where each stage may take a different amount of time to complete, a multicycle implementation potentially allows up to N times speedup over a single cycle implementation if there are instructions that must pass through only one stage.
    d. In a multicycle implementation where every stage takes the same amount of time to complete, the instruction that passes through the most stages will have the same execution time as in a single-cycle implementation.
    e. In a multicycle implementation where every stage may take a different amount of time to complete, the instruction that takes the longest will NOT have the same execution time as in a single cycle implementation.

5. We wish to implement a no-operation (NOP) instruction in the MIPS datapath that, when executed, effectively does nothing except waste time (this sounds strange but is actually very useful in applications that must meet strict minimum timing requirements). Choose all the signal combinations that would correctly implement NOP (note: X = Don't care.). NOP is implemented as an R-format instruction with a function code of 0x3F. (Note: Here we only consider the specifications for the signal values; we do not consider actual implementation yet).

    a. RegDst=1, RegWrite=0, AluSrc=0, MemWrite=0, MemRead=0, MemToReg=0, PCSrc=0
    b. RegDst=1, RegWrite=X, AluSrc=X, MemWrite=0, MemRead=0, MemToReg=X, PCSrc = 0
    c. RegDst=0, RegWrite=0, ALuSrc=1, MemWrite=0, MemRead=0, MemToReg=1, PcSrc=1
    d. RegDst=X, RegWrite=0, AluSrc=X, MemWrite=X, MemRead=0, MemToReg=X, PCSrc=0
    e. RegDst=X, RegWrite=0, AluSrc=X, MemWrite=0, MemRead=0, MemToReg=1, PCSrc=0

MCQ Questions

Each question has exactly one correct answer and is worth 1 mark.

Section 1 – Number Systems

6. To find the 10's complement of an N-digit decimal number Y, we do $10^N - Y$. Which ONE of the following statements is TRUE?

    a. In an N-digit 10's complement number system, the leftmost digit has a weight of 10^N.
    b. In an N-digit 10's complement number system, the leftmost digit has a weight of 10^{N-1}**.**
    c. In an N-digit 10's complement number system, the leftmost digit has a weight of -10^N (negative 10^N)
    d. In an N-digit 10's complement number system, the leftmost digit has a weight of -10^{N-1} (negative 10^{N-1}).
    e. None of the choices in this question (except this one) are true.

7. In a 4-bit 1's complement number system, doing -3 - 6 (negative 3 minus 6) will give us:

    a. -9
    b. 5
    c. 6
    d. -5
    e. -6

Section 2 – C Programming

8. What does the following C function do? Assume that a long is twice the size of an int.

```
unsigned long mystery(unsigned int x, unsigned int y) {
   unsigned long a = 0, b = x;

   while(y) {
               if(y & 0b1) {
                           a = a + b;
               }

               b = b << 1;
               y = y >> 1;
   }

   return a;
}
```

a. Adds x and y.
b. Shifts b to the right by 1 bit and y to the left by 1 bit and adds them together.
c. Produces an even parity for x and y.
d. Multiplies x and y.
e. Divides x by y.

9. What does the following C function do?

```
unsigned char mystery2(unsigned char x, unsigned char y) {
    unsigned char a = x;
    unsigned char b = y;
    unsigned char c = 1;
    unsigned char d = 0;

    while(c) {
        if((a & 1) != (b & 1)) {
            d = d + c;
        }

        a = a >> 1;
        b = b >> 1;
        c = c << 1;
    }

    return d;
}
```

a. Shifts a and b left by 1 bit and c right by 1 bit, then adds c to d.
b. Does a bit-wise XOR of x and y.
c. Checks whether both x and y are equal.
d. The function will never exit.
e. Does a bit-wise division of x and y.

Section 3. Assembly Language

Dueet wrote the following code in MIPS assembly to process some elements of an array whose base address is in $s2.

1		addi $s0, $zero, 0
2		addi $s1, $zero, 0
3		addi $t0, $s2, 0
4		addi $t1, $s2, 40
5	x:	slt $t2, $t0, $t1
6		beq $t2, $zero, z
7		lw $t3, 0($t0)
8		addi $t0, $t0, 4
9		andi $t3, $t3, 1
10		beq $t3, $zero, y
11		addi $s0, $s0, 1
12		j x
13	y:	addi $s1, $s1, 1
14		j x
15	z:	

10. What does this code do?

    a. It counts the number of array elements divisible by 2 and number of items divisible by 3 and stores the results in $s0 and $s1 respectively.
    b. It counts the number of array elements with even parity and odd parity and stores the results in $s0 and $s1 respectively.
    c. It counts the number of odd and even array elements and stores the results in $s0 and $s1 respectively.
    d. It counts the number of array elements with even parity and odd parity and stores the results in $s1 and $s0 respectively.
    e. It counts the number of array elements that are zero and non-zero and stores the results in $s0 and $s1 respectively.

11. Dueet's software license for her assembler has expired, and she decided to "hand-assemble" the code above by consulting the datasheet and translating the assembly code into machine code. However when she ran her code she found that her code always terminates with $s1=0 regardless of the contents of the array, though $s0 is correct. What is the likeliest problem?

    a. The j statement at line 12 was encoded to jump to line 2 instead of line 5.
    b. She calculated the offset for the branch instruction at line 10 based on the instruction at line 10 instead of line 11.
    c. She calculated the offset for the jump statement at line 14 using the address at line 15 instead of the address at line 13.
    d. She encoded the andi instruction at line 9 to do "andi $t3, $t3, 0" instead of "andi $t3, $t3, 1".
    e. She calculated the offset for the branch at line 10 based in bytes rather than words.

Section 4. Fill In The Blanks (Expanding Opcodes)

In this section we assume a processor with a fixed 16-bit instruction length, 16 registers, and the following 3 instruction classes:

Class A: Two registers.
Class B: One register.
Class C: One register, one 8-bit immediate value.

In all cases we assume that the encoding space for opcodes is fully utilized.

12. The <u>minimum</u> number of opcodes that can be encoded on this machine is _____.

13. The <u>maximum</u> number of opcodes that can be encoded on this machine is _____.

Section 5. Fill In The Blanks (Floating Point Numbers):

We have a 16-bit floating point number system with the following format:

1 sign bit for the mantissa	6 bit exponent in 2's complement. No reserved bit patterns with special meanings.	9 bit normalized mantissa, no hidden bit, in sign and magnitude representation.

The leftmost bit and the rightmost 9 bits form a sign-and-magnitude mantissa. Answer the following questions, filling in the **exact** numeric answer each time.

14. Within the representation range of this number system (i.e. between the most negative and most positive numbers that can be represented), the number _____ cannot be represented.

15. The most negative number that can be represented in this number system is _____.