3NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

MIDTERM TEST

AY2024/25 Semester 1

CS2100 – COMPUTER ORGANIZATION

DURATION OF ASSESSMENT: 1.5 HOURS

QUESTION PAPER

Instructions (please read carefully):

1. This assessment consists of **28** questions printed on **TEN (10)** pages, including this cover page.

2. This is an **OPEN BOOK** assessment.

3. You are to write your answers on the single A4 Answer Sheet provided. We will collect ONLY your Answer Sheet at the end of the test. You may keep this question paper.

4. Do not attach any additional sheet of papers to the Answer Sheets, it will jam our scanning.

5. **All** questions **must** be answered in the space provided on the Answer Sheet; no extra sheets will be accepted as answers. Any writing at the back of the page or outside the designated answer area will not be considered. Note that the questions do not carry the same marks.

6. You are allowed to bring an approved calculator, but it cannot have any form of external communication capability.

7. You are allowed to write with pencils, as long as it is legible. Marks may be deducted for unrecognisable handwriting.

8. Hexadecimal strings are written with the "`0x`" prefix. All addresses are byte addresses unless otherwise specified.

9. The last sheet in this question paper is intentionally left blank (on both sides). You may use it as scratch for your working. It will not be collected or graded.

Part A: Multiple Choice Questions (25 marks)

Each MCQ has only one correct answer. Shade the corresponding circle on the Answer Sheets for your answer using a pencil, ensuring that the shading is prominent and the other options are not tainted. No other form of marking out your answer will be considered. We are using machine recognition to grade the MCQs.

Question 1

What is the result of the following subtraction in 8-bit signed magnitude representation:

$$00110010_{sm} - 10111101_{sm}$$

A. 10010000
B. 10010001
C. 01101111
D. 11101110
E. None of the above

Question 2

Bob decided to design a new processor that works with 137 bits integers. He decided also to use two's complement to represent signed integer. Which of the following in hexadecimal is the largest *positive* integer that can exist in his representation system?

A. 0x0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
B. 0x1FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
C. 0x3FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
D. 0x7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
E. None of the above

Question 3

This is related to Assignment 1 Question 2. Consider the following:

$$3a79_X - 2035_Y = 1792_Z$$

Which one of the following is **not** a valid solution?

A. $X = 12$, $Y = 13$ and $Z = 11$
B. $X = 15$, $Y = 16$ and $Z = 14$
C. $X = 11$, $Y = 12$ and $Z = 10$
D. $X = 13$, $Y = 15$ and $Z = 11$
E. None of the above

Question 4

Which of the following in hexadecimal would represent the smallest positive normalized floating point number representable in the IEEE Standard 754 single precision (32-bit) floating point?

A. 0x00000001
B. 0x00800000
C. 0x00800001
D. 0x01000000
E. None of the above

Question 5

What value does the hexadecimal 0xBEA00000 represent in the IEEE Standard 754 single precision (32-bit) floating point?

A. 0.12565
B. -0.12565
C. 0.5125
D. -0.3125
E. None of the above

Question 6

Which of the following is the smallest positive base-10 value that is representable as a 32-bit integer but cannot be represented precisely as in IEEE Standard 754 single precision (32-bit) floating point? In other words, when one converts the said value (let's call it A) into the IEEE Standard 754 single precision (32-bit) floating point representation (let's call it F), and then convert F back into an integer (let's call it B), then A will not be equal to B.

A. 16777215_{10}
B. 16777216_{10}
C. 16777217_{10}
D. 16777218_{10}
E. None of the above

Question 7

Which of the following is a valid C initialization statement (i.e., no compilation warning or error, and code will run accordingly)?

A. `int A[4] = {1, 2, 3, 4, 5};`
B. `int A[4] = {1, 2, 3};`
C. `int A[4] = [1, 2, 3, 4];`
D. `int A[4] = {1, , 3, 4};`
E. None of the above

Question 8

Consider the following C program:

```
for (int i=0; i<10; i++) {
    printf("%d ", ++i);
}
```

What output would be printed out at the terminal after compiling and executing this C loop?

A. `1 2 3 4 5`
B. `0 2 4 6 8`
C. `1 3 5 7 9`
D. `1 2 3 4 5 6 7 8 9`
E. None of the above

Question 9

Consider the following C program:

```
#include <stdio.h>

int x = 1;

int main(int argc, char *argv[])
{
    int x = 2;

    for (int x=0; x<4; x++) {
        int x = 3;
        printf("%d ", x++);
    }

    printf("%d\n", x);
}
```

What output would be printed out at the terminal after compiling and executing this program?

A. 3 3 3 3 2
B. 3 3 3 3 3
C. 3 4 5 6 7
D. 0 3 4 6 8
E. None of the above

Question 10

Suppose a small modification is made to the C program from Question 9 (underlined below):

```
#include <stdio.h>

int x = 1;
int main(int argc, char *argv[])
{
    int x = 2;

    for (int x=0; x<4; x++) {
        static int x = 3;
        printf("%d ", x++);
    }
    printf("%d\n", x);
}
```

What output would be printed out at the terminal after compiling and executing this program?

A. 3 3 3 3 4
B. 3 4 5 6 7
C. 0 1 2 3 4
D. 3 4 5 6 2
E. None of the above

Question 11

Consider the following C program:

```
#include <stdio.h>

int A[4] = {1, 2, 3, 4};

int main(int argc, char *argv[])
{
    int x, y;
    int *p[2], **q;

    x = A[0];
    y = A[1];

    p[0] = &(A[1]);
    p[1] = &(A[3]);
    q = p;

    *q++ = &(A[2]);
    **q = 100;

    printf("%d %d %d %d\n", A[0], A[1], A[2], A[3]);
}
```

What output would be printed out at the terminal after compiling and executing this program?

A. 1 2 3 4
B. 1 2 3 100
C. 1 2 100 4
D. 1 100 3 4
E. None of the above

Question 12

What is the hexadecimal encoding for the following instruction:

```
addi $t1, $sp, -120
```

A. **0x23A90088**
B. **0x23A9FF88**
C. **0x239AFF88**
D. **0x293A0088**
E. None of the above

Question 13

Give the hexadecimal encoding for the following branch instruction:

```
bne $a0, $t8, EXIT
```

Assume that this instruction is at PC = **0x401C** and **EXIT** is at PC = **0x35F0**.

A. **0x140035F0**
B. **0x1400401C**
C. **0x1498FD74**
D. **0x1498FD75**
E. None of the above

Question 14

As a standard practice, we use an **ori** instruction to set a register with the 16-bit immediate zero extended to 32 bits, after a **lui** instruction has loaded the upper 16 bits. Which of the following MIPS instructions (and they are all legitimate ones) can also be used instead of the **ori** instruction to do the same thing?

A. **addi**
B. **andi**
C. **slti**
D. **nor**
E. None of the above

Question 15

Consider the following MIPS instruction given in hexadecimals:

```
0x08012348
```

Assuming that the PC of this instruction is **0x2FFFFFFC**, what would be the PC after the execution of this instruction?

A. **0x20012348**
B. **0x2F048D20**
C. **0x30048D20**
D. **0x3001234C**
E. None of the above

Question 16

According to the MIPS reference data sheet, the last occupied location on the top of the stack is pointed to by **`$sp`** and is word aligned. Which of the following would implement a pseudo stack push instruction that pushes the content of $x onto the stack?

A.
```
sw $x, -4($sp)
```

B.
```
addi $sp, $sp, -4
sw $x, 0($sp)
```

C.
```
sw $x, -4($sp)
addi $sp, $sp, 4
```

D.
```
addi $sp, $sp, -4
lw $x, 0($sp)
```

E. None of the above

For **Questions 17 and 18**, we will assume the parameters used in the standard MIPS encoding as shown in the MIPS Reference Sheet. In particular, we will assume that there are three instruction types: R-type, I-type, and J-type instructions. We will assume that for R-type, there are two subtypes – R-type integer and R-type floating point instructions that have the opcode of **`0x00`** and **`0x11`**, respectively.

Question 17

Given the above setup, how many distinct R-type integer instructions can be encoded?

A. 32
B. 64
C. 128
D. 256
E. None of the above

Question 18

Suppose now we give up on having the shift instructions being of R-type and instead make them I-type, thus freeing up the bits used to encode the shift amounts. Further, suppose we use these free slots for integer instructions while keeping the R-type floating point operations unchanged. Now how many distinct R-type integer instructions can we encode?

A. 256
B. 1024
C. 2048
D. 4096
E. None of the above

Question 19

Consider a 16-bit fixed length instruction set. Suppose there are three types of instructions:

- Type A: 3 operands, 3 bits each.
- Type B: 2 operands, 3 bits each.
- Type C: 1 3-bit long operand
- There must be at least one distinct instruction of each type.

What is the maximum number of instructions that can be formed using this specification?

A. 3062
B. 3584
C. 8076
D. 8114
E. None of the above

Question 20

What is the minimum number of instructions that can be formed using the exact specification as in Question 19, and all opcodes are used?

A. 126
B. 132
C. 231
D. 4867
E. None of the above

Question 21

In the MIPS datapath taught in class, which of the following statements is false?

A. The datapath to RR2 consists of 5-bits as you have 32 registers in total.
B. RD2 is used by all instructions.
C. The ALU is a combinational circuit.
D. In the 1-bit ALU, C_{in} is only used for the + operation.
E. `$zero` is a register that always returns the value zero and has no effect when it is written to.

Question 22

In the MIPS datapath taught in class, let us assume that `ALUoutput` is the output from the ALU. In the following, we check the value of `MemToReg` and then accordingly write to the register file.

```
WriteData = MemToReg ? Memory[ALUoutput] : ALUoutput
```

If we change the above datapath to the following:

```
WriteData = ALUoutput
```

which one of the following instructions would no longer work?

A. **`sw $s1, 2($s3)`**
B. **`addi $s1, $s3, -50`**
C. **`lw $s1, 2($s3)`**
D. **`j 0x12345678`**
E. None of the above

Question 23

In the MIPS datapath taught in class, which of the following statements is true?

A. The PC is a special register that is 36-bits long.
B. **`ALUcontrol`** has 6-bits as its input and 4-bits as its output.
C. **`ALUop`** is responsible for controlling the control signal **`RegDst.`**
D. The PC is updated on the rising edge of the next clock cycle.
E. The PC is updated on the falling edge of the next clock cycle.

Question 24

In the MIPS datapath taught in class, why are the inputs to the **`MemToReg`** multiplexer reversed?

A. This is so that there is some variety in the datapath.
B. The **`MemToReg`** multiplexer is the only one that deals with memory.
C. This is so that the wires do not cross over each other in the diagram.
D. It is because for the `sw` instruction, the source and destination registers are swapped
E. None of the above.

Question 25

In the MIPS datapath taught in class, which of the following statements is false?

A. MIPS has a total of 32 registers.
B. MIPS opcodes are all 6 bits long.
C. The **`funct`** field is sometimes used to set the **`ALUcontrol`** signal.
D. The **`ALUop`** signal for the **`beq`** instruction is **`01.`**
E. None of the above.

Part B: Short questions (3 questions, 5 marks each)

You must answer in the space provided in the Answer Sheets. Any writing outside the answer box will be disregarded.

Question 26

Suppose we do not have a real MIPS lui instruction. Instead, it is a pseudo-instruction of the form "**`lui $x, <16-bit const>`**", where "**`$x`**" is any of the valid registers (the assembler obtains the actual number) and the constant is 16-bit. We need to implement it using (the remaining) real MIPS instructions. The assembler uses a text rewriting process not unlike C macros. What you need to do is write a text template. Use "$x" and "<16-bit const>" to represent the target register and the 16-bit constant in the original **`lui`** that the assembler will use it to instantiate an instant from your template and replace the line where lui is in the code with the instance. Show what your template looks like. Don't worry about style, the solution is to test the concept – though you have to be clear in your description, and your assumptions has to be realistic. You do have to be careful that your code must work in all code circumstances and not compromise values in the registers.

Question 27

Suppose we want to design a new 32-bit fixed instruction set that is inspired by the MIPS encoding scheme. There are three types of instructions, i.e., R-type, I-type, and J-type. Now suppose the number of general purpose registers is increased to 64. Assuming that

- We do shifts using I-type instructions, thus doing away with the **`shamt`** field;
- The opcode field must be present but need not be 6 bits long.
- R-type instructions still has a **`func`** field whose length need not be 6;
- The immediate field is still 16-bit two's complement;
- There is at least ten R-type instruction;
- There is at least ten I-type instruction;
- We only need exactly two J-type instructions. The encoding is the same as in MIPS but the number of bits to be taken from the upper part of PC need not be 4. The address field should be as long as possible, requiring as few bits from the upper part of the PC as possible.

Design an expanding opcode scheme that **maximizes** the total number of instructions. Describe your design and fill in the numbers that are the result of your design in the respective boxes given.

Question 28

Consider the MIPS datapath covered in class. For the MIPS instruction encoded as **0x10000103** fill in the corresponding elements in the boxes on the Answer Sheets. Use the notation $R to represent register number R, [$R] to represent the content of register number R and Mem(X) to represent the memory data at address X. Assume the PC value is 0x491 at the start .

== END OF PAPER ==