NATIONAL UNIVERSITY OF SINGAPORE

CS2100 – COMPUTER ORGANISATION

(Semester 2: AY2019/20)

Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. Please write your Student Number on the **ANSWER BOOKLET**. Do not write your name.
2. This assessment paper consists of **EIGHT (8)** questions and comprises **NINE (9)** printed pages.
3. This is an **OPEN BOOK** assessment.
4. Answer all questions and write your answers in the **ANSWER BOOKLET** provided.
5. Make sure your answers are clearly written/typed.
6. You are to submit only the **ANSWER BOOKLET** and no other document.

1. **[16 marks]**
Study the following MIPS program:

```
    addi $s1, $zero, 0
L:  andi $t0, $s0  , 0xF  # $s0 initialized as 0xBEBEC0C0
    add  $s1, $s1  , $t0
    srl  $s0, $s0  , 4
    bne  $s0, $zero, L
```

(a) What is the value of register `$s1` after we finish running the program with `$s0` initialized as `0xBEBEC0C0`? Write your answer in ***hexadecimal***. [2 marks]

(b) What is the maximum possible value of `$s1`? What is the initial value of `$s0` that can give this maximum possible value of `$s1`? Write your answers in ***hexadecimal***. [4 marks]

(c) Write the equivalent C code for the MIPS code above. You should use registers as your variable names (e.g., `$t1 = 5;`) [6 marks]

(d) Give the encoding of the instruction `srl $s0, $s0, 4` in ***hexadecimal***. [2 marks]

(e) Give the encoding of the instruction bne `$s0, $zero, L` in ***hexadecimal***. [2 marks]

2. **[4 marks]**
Convert -1.875 into IEEE-754 single-precision floating-point representation.
Write your answer in ***hexadecimal***. [4 marks]

3. **[10 marks]**
von Neumann architecture stores both data and code in the same memory. Therefore, we can have a program (such as a compiler) to process other programs. Here, we want to test your understanding of MIPS by writing MIPS instructions to reason about MIPS instructions.

(a) Consider any MIPS instruction stored in `$s0`. Using the minimum number of MIPS instructions, set the value of register `$t0` to 1 if the instruction is an R-format instruction, otherwise set the value to 0. [3 marks]

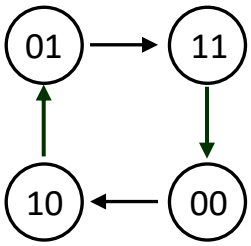
(b) Consider an encoding of an R-format MIPS instruction stored in `$s1`. Using the minimum number of MIPS instructions, extract the field `rs` into register `$t1`. [3 marks]

(c) Consider an opcode stored in `$s2`. Write a sequence of MIPS instructions to generate the value of ALUop control signal and store it in `$t2`. The table below shows the value of ALUop for each instruction type. You may assume that the opcode will only be for `lw`, `sw`, `beq`, or any R-format instructions. [4 marks]

Instruction Type	**ALUop**
`lw / sw`	`00`
`beq`	`01`
`R-type`	`10`

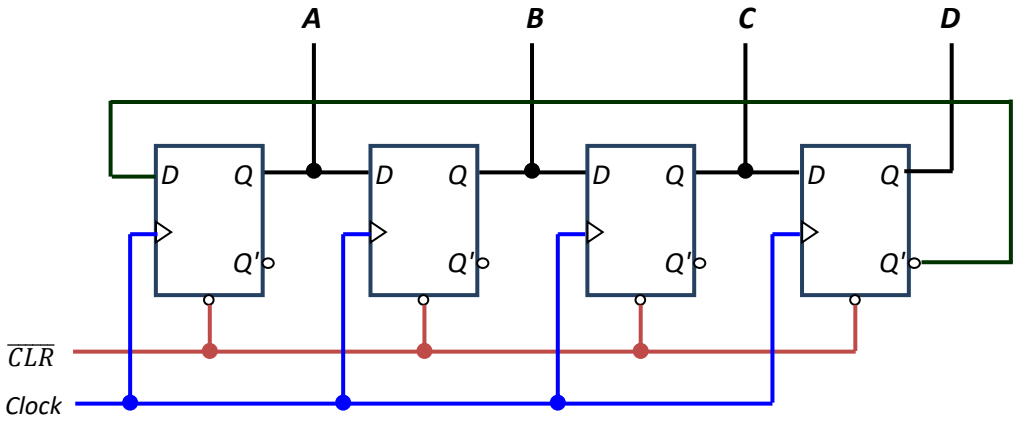
4. **[12 marks]**

(a) Implement a sequential logic circuit that cycles through the following states using only *JK* flip-flops. The states are represented by 2-bit values *AB*.



Write out the **simplified SOP expressions** for all the flip-flop inputs. [4 marks]

(b) Study the sequential circuit below, which uses four *D* flip-flops. The $\overline{CLR}$ input is an asynchronous input that is used to clear the value of a flip-flop. Note that *Q'* of the right-most flip-flop is connected to the *D* input of the left-most flip-flop.



The circuit is initialised to *ABCD* = 0000 by clearing all flip-flops to zero (i.e. the *Q* output for every flip-flop is cleared to zero). This sequential circuit cycles through a number of states. The first state (call it state 0) is 0000, the second state (call it state 1) is 1000.

(i) How many states are there altogether? List out the states in sequence, i.e. 0000 → 1000 → … [4 marks]

(ii) Suppose the function State(0) is true when the circuit is in state 0, and false otherwise. The simplest Boolean expression for State(0) is *A'*·*D'*.

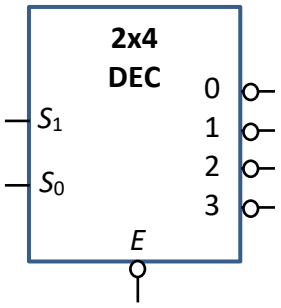
Similarly, State(1) is true when the circuit is in state 1 or false otherwise; and State(2) is true when the circuit is in state 2 or false otherwise. What are the simplest Boolean expressions for State(1) and State(2)? [4 marks]

5. **[12 marks]**
For the parts below, you are to assume that logical constants 0 and 1 are available but complemented literals are not available.

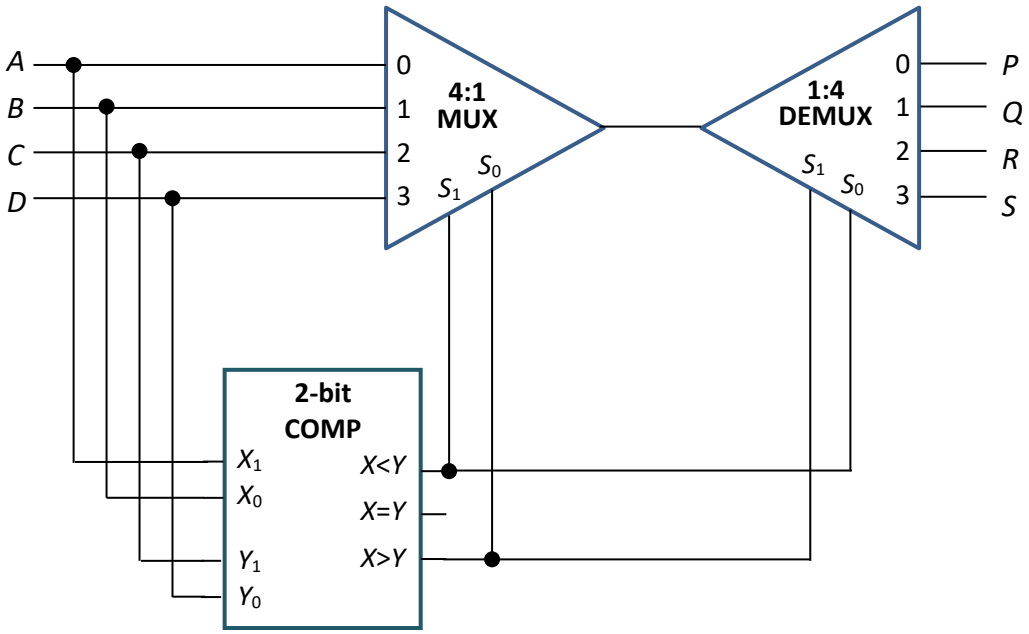
(a) A device **NonZero** takes in a 3-bit unsigned number *ABC*. Its output, *N*, is 0 if the value represented by *ABC* is 0, or 1 if the value represented by *ABC* is not zero.

(i) Write the simplified SOP expression for *N*. [2 marks]

(ii) Implement *N* using the following single 2x4 decoder with zero-enable and negated outputs, with no other logic gates and devices. [4 marks]



(b) The circuit below uses a 2-bit magnitude comparator, a 4:1 multiplexer and a 1:4 demultiplexer. Inputs are *A*, *B*, *C*, *D* and outputs are *P*, *Q*, *R*, *S*. Write the simplified SOP expressions for *P*, *Q* and *R*. [6 marks]



6. **[12 marks]**

For the parts below, you are to assume that logical constants 0 and 1 are available but complemented literals are not available. Note also that a circuit that is correct but uses more logic gates or devices than necessary will be given only partial credit.

The following block diagrams are referred to and explained in the question.

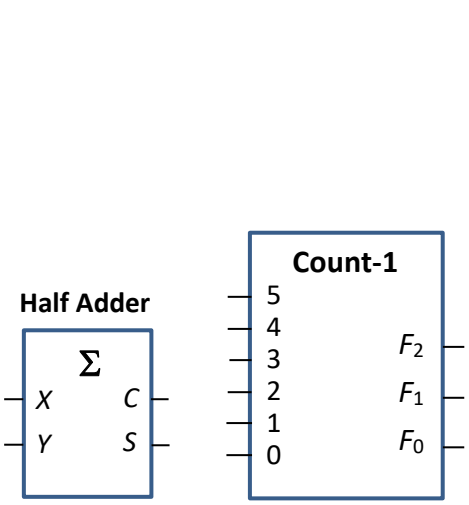


Figure 6(a)

Figure 6(b)

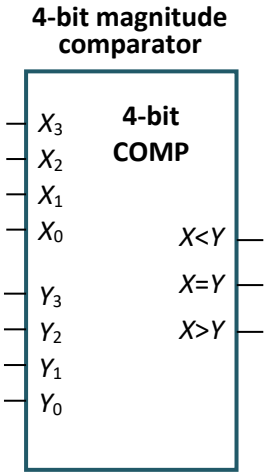


Figure 6(c)

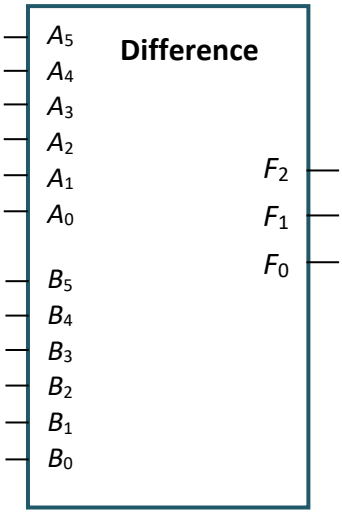


Figure 6(d)

(a) Design a circuit to take in two 6-bit codes $P=P_5P_4P_3P_2P_1P_0$ and $Q=Q_5Q_4Q_3Q_2Q_1Q_0$, and generate as output $D_2D_1D_0$ the number of pairwise bits in P and Q that are different. For example, P=001001 and Q=010100 differ by 4 bits ($P_4 \neq Q_4$, $P_3 \neq Q_3$, $P_2 \neq Q_2$, $P_0 \neq Q_0$), so $D_2D_1D_0$=100; P=111110 and Q=011111 differ by 2 bits ($P_5 \neq Q_5$, $P_0 \neq Q_0$), so $D_2D_1D_0$=010; and 101010 and 010101 differ by 6 bits, so $D_2D_1D_0$=110.

Recall that we have designed a circuit in class to count the number of ones in a 6-bit number. The block diagram of the circuit, called a **Count-1 device**, is shown in Figure 6(b). If the input 001001 is fed into the Count-1 device, the output would be 010 (two).

Using only half adders (Figure 6(a)) and one **Count-1 device** with no other logic gates or devices, design a circuit to count the number of pairwise bit differences of two 6-bit codes P and Q. [3 marks]

(b) A certain 6-bit code $A_5A_4A_3A_2A_1A_0$ contains only eight valid code values shown in the table below.

$A_5A_4A_3A_2A_1A_0$
0 0 0 1 1 0
0 1 1 0 0 0
0 1 1 0 0 1
0 0 0 1 1 1
1 0 0 1 1 1
1 1 1 0 0 1
1 1 1 0 0 0
1 0 0 1 1 0

You are to design a logic circuit for Boolean function $V(A_5,A_4,A_3,A_2,A_1,A_0)$ that returns 1 if the input $A_5A_4A_3A_2A_1A_0$ is one of these eight valid values, or returns 0 otherwise.

Your circuit should contain the fewest number of half adders and at most one 4-bit magnitude comparator (Figure 6(c)), and no other logic gates or devices. [5 marks]

(c) Referring to the same code given in part (b) above.

You are to pick two code values from this table and send them to some device for processing, but before that, you want to check that these two code values indeed come from this table. The two code values you pick could be the same value.

Design a logic circuit for Boolean function $E(A_5,A_4,A_3,A_2,A_1,A_0,B_5,B_4,B_3,B_2,B_1,B_0)$ where $A=A_5A_4A_3A_2A_1A_0$ and $B=B_5B_4B_3B_2B_1B_0$ are two 6-bit values. The function E returns 1 to indicate that values A and B certainly cannot be from the table. However, if E returns 0, it is non-conclusive and further check is required.

You are to design this circuit using the device designed in part (a), which is called the **Difference** device, whose block diagram is shown in Figure 6(d), and a single 4-bit magnitude comparator, and no other logic gates or devices. [4 marks]

7. **[14 marks]**

An image consisting of 1024×1024 pixels are stored in a one-dimension integer array with 2^{20} elements in **row-major order**. Each element is a 32-bit integer, consisting of 4 parts: *r*, *g*, *b*, *t*, representing red, green, blue and transparency. The values of red, green, blue and transparency occupy 8 bits and are unsigned numbers in the range 0 through 255.

Given two images stored in integer arrays *A* and *B*, the following algorithm creates a new image in array *C* such that the RGB (red, green, blue) components of every pixel in image *C* is the exclusive-or value of the RGB components of the corresponding pixels in images *A* and *B*, and the transparency of the pixel in image *C* is the average of the transparency of the corresponding pixels in images *A* and *B*.

```
for (int r = 0; r < 1024; r++)
   for (int c = 0; c < 1024; c++)
      Cimage[r][c] = merge(Aimage[r][c], Bimage[r][c]);
```

For example, if the first pixel of image *A* is pure red with transparency value of 125 (or 0xFF00007D), and the first pixel of image *B* is pure blue with transparency value of 60 (or 0x0000FF3C), then the first pixel of image C will be purple with transparency of 92 (or 0xFF00FF5C).

The MIPS code fragment is given on the next page. Assuming a 5-stage MIPS pipeline we studied in class, and considering only the code segment from Inst1 (**`add $t1,$0,$0`**) to Inst23 (**`addi $t2,$t2,1`**) inclusive, answer the following questions. You need to count until the last stage of Inst23.

a. How many cycles does this code segment take to complete its execution in the first iteration in an ideal pipeline (that is, with no delays)? [1 mark]

For parts (b)–(d) below, given the assumption for each part, how many <u>additional</u> cycles does this code segment (Inst1 to Inst23 inclusive) take to complete its execution in the first iteration compared to an ideal pipeline? (For example, if part (a) takes *X* cycles and part (b) takes *Y* cycles, you are to answer part (b) with the value *Y* – *X*.)

b. Assuming <u>without forwarding and branch decision is made at MEM stage</u> (stage 4). No branch prediction is made and no delayed branching is used. [3 marks]

c. Assuming <u>without forwarding and branch decision is made at ID stage</u> (stage 2). No branch prediction is made and no delayed branching is used. [3 marks]

d. Assuming <u>with forwarding and branch decision is made at ID stage</u> (stage 2). Branch prediction is made where the branch is predicted not taken, and no delayed branching is used. [3 marks]

e. Assuming the condition in part (d) but now we want to implement delayed branching as well. Is it possible to fill in the branch-delay slot for each of the two **`beq`** instructions? If possible, indicate which instructions can be used to fill in the slots; if not possible, explain why. [2 marks]

f. For branch prediction, we may predict either the branch is taken or not taken. Which of the two choices is easier to implement, and why? [2 marks]

You may assume that $s0, $s1, $s2 and $s3 have been initialised.

```
        # $s0 = 1024
        # $s1 = base address of array A
        # $s2 = base address of array B
        # $s3 = base address of array C
        add  $t1, $0,  $0       # Inst1 : $t1 = r = 0
        add  $t2, $0,  $0       # Inst2 : $t2 = c = 0
        add  $t3, $0,  $0       # Inst3 : $t3 = element index

L1:     slt  $t0, $t1, $s0      # Inst4 : is r < 1024?
      2 beq  $t0, $0,  E1       # Inst5 : done with rows

L2:   3 slt  $t0, $t2, $s0      # Inst6 : is c < 1024?
      2 beq  $t0, $0,  E2       # Inst7 : done with columns

      3 add  $t4, $s1, $t3      # Inst8 : $t4 = addr of A[r][c]
      2 lw   $t5, 0($t4)        # Inst9 : $t5 = A[r][c]
        add  $t6, $s2, $t3      # Inst10: $t6 = addr of B[r][c]
      2 lw   $t7, 0($t6)        # Inst11: $t7 = B[r][c]

        # Inst12-19 to create image C from A and B
      2 xor  $t9, $t5, $t7      # Inst12
        andi $t5, $t5, 0xFF     # Inst13
        andi $t7, $t7, 0xFF     # Inst14
      2 add  $t0, $t5, $t7      # Inst15
      2 srl  $t0, $t0, 1        # Inst16
        srl  $t9, $t9, 8        # Inst17
      2 sll  $t9, $t9, 8        # Inst18
      2 or   $t9, $t9, $t0      # Inst19

        add  $t8, $s3, $t3      # Inst20: $t8 = addr of C[r][c]
      2 sw   $t9, 0($t8)        # Inst21: C[r][c] = $t9

        addi $t3, $t3, 4        # Inst22: next element
Inst23 → addi $t2, $t2, 1       # Inst23: increment c
        j    L2                 # Inst24

E2:     add  $t2, $0,  $0       # Inst25: reset c to 0
        addi $t1, $t1, 1        # Inst26: increment r
        j    L1                 # Inst27
E1:
```

8. **[20 marks]**
Refer to the same MIPS code in the previous question.

We assume that the cache is used for **lw** instructions but not for **sw** instructions so you may ignore array *C* for this question.

Integer arrays *A* and *B* are stored starting at memory addresses 0x**0040CCC0**, and 0x**000002C0** respectively. Each integer occupies one word which is 4 bytes long. Recall in the previous question that the arrays are stored in **row-major order** of the images they represent.

For parts (a), (b), (c): Given a **direct-mapped data cache** with 64 words in total and each block contains 4 words.

a. How many bits are there in the index field? In the byte offset field? [2 marks]

b. Which cache block is *A*[0] mapped to? Which block is *B*[60] mapped to? Write your answer in decimal. [2 marks]

c. What is the cache hit rate for array *A*? For array *B*? [2 marks]

For parts (d), (e), (f): Given a **two-way set associative data cache** with 64 words in total and each block containing 4 words. LRU (least recently used) algorithm is used for replacement.

d. Which cache set is *A*[0] mapped to? Which set is *B*[60] mapped to? Write your answer in decimal. [2 marks]

e. What is the cache hit rate for array *A*? For array *B*? [2 marks]

f. Suppose the MIPS code is unchanged but the images are stored in **column-major order** for their corresponding arrays, what would be the cache hit rates for arrays *A* and *B*? Explain. [2 marks]

For parts (g), (h), (i): Given a **direct-mapped instruction cache** with 16 words in total and each block contains 4 instructions (words). The first instruction (`add $t1, $0, $0`) is at memory address 0x**04FFFFF8**.

We consider only the inner loop "`for (int c=0; c<1024; c++)`" here.

g. On the instruction cache diagram on the Answer Booklet, fill in the location where the first instruction (call it Inst1) will be loaded into. [2 marks]

h. How many misses are there in the 1st iteration (Inst1 to Inst24 inclusive)? [3 marks]

i. How many misses are there in the 2nd iteration (Inst6 to Inst24 inclusive)? [3 marks]

~~~ END OF PAPER ~~~