NATIONAL UNIVERSITY OF SINGAPORE

CS2100 – COMPUTER ORGANISATION

(Semester 1: AY2021/22)

Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This assessment paper consists of **SIXTEEN (16)** questions (excluding question 0) in **THREE (3)** parts and comprises **FOURTEEN (14)** printed pages.
2. Answer ALL questions.
3. This is an **OPEN BOOK** assessment.
4. The maximum mark of this assessment is 100.

Question	Max. mark
Q0	3
Part A: Q1 – 6	12
Part B: Q7 – 11	15
Part C: Q12	12
Part C: Q13	13
Part C: Q14	13
Part C: Q15	14
Part C: Q16	18
Total	**100**

5. You are to submit a single pdf file (size $\leq$ 20MB) to your submission folder on LumiNUS.
6. Your submitted file should be named after your Student Number (eg: A1234567X.pdf) and your Student Number should be written on the first page of your file.
7. Do NOT write your name anywhere in your submitted file.

——— END OF INSTRUCTIONS ———

You do not need to write any answer for question 0.

0. (a) Submit a **single pdf file** into the submission folder. [1 mark]

   (b) Name your pdf file with your Student Number (eg: A1234567X.pdf). [1 mark]

   (c) Write your Student Number (not your name!) on the first page of your file. [1 mark]

Part A: Multiple-Choice Questions [Total: 6×2=12 marks]

Each multiple-choice question (MCQ) is worth **TWO marks** and has exactly **one** correct answer. You may write your answers into the boxes in the Answer Sheets provided or if you are using your own paper, write your answers on a **single line** to conserve space. For example:

1. A 2. B 3. C 4. D …

Please write in **CAPITAL LETTERS**.

1. If $(1022)_3 = (50)_x$, what is x?

   A. 4
   B. 5
   C. 6
   D. 7
   E. None of the above.

2. What is the output of the following C program?

```
#include <stdio.h>
void foo(int *i){
    (*i)++;
}

int main(){
    int i = 20;
    int *ptr = &i;
    foo(ptr);
    printf("i = %d\n", i);
    return 0;
}
```

   A. `i = 20`
   B. `i = 21`
   C. `i = 22`
   D. No output; runtime error.
   E. No output; compilation error.

3. What is the output of the following C program?

```
#include <stdio.h>
int main(){
    int i = 100;
    int *ptr = &i;
    *ptr += 1;
    printf("%d %d\n", *ptr, i);
    return 0;
}
```

A. `101 101`
B. `100 101`
C. `101 100`
D. `100 100`
E. None of the above.

4. For the **lw $20, 100($10)** instruction, which one of the following is true?

A. Data stored in the RR1 from register file and Write Data from Data Memory are not used.
B. Data stored in the RR2 and WD from register files are not used.
C. Data stored in the RR2 from register file and Write Data from Data Memory are not used.
D. Data stored in the RR2 and WR from register file are not used.
E. None of the above.

5. Given the Boolean function $F(A,B,C,D) = \Sigma m(3,10,11,15) + \Sigma x(1,2,4,7,9,12,14)$ where *x* denotes don't-care, how many EPIs (essential prime implicants) are there on the K-map of *F*?

A. 0
B. 1
C. 2
D. 3
E. 4.

6. For the following MIPS code, what is the equivalent C code? Use the following mapping of variables and registers: f, g, h, i, and j are mapped to registers $s0, $s1, $s2, $s3, and $s4 respectively.

```
Loop1:  bne $s3, $s4, Loop2
        add $s0, $s1, $s2
        j   Loop3
Loop2:  sub $s0, $s1, $s2
Loop3:
```

What does the code do?

A.
```
if (i==j) f = g+h;
f = g-h;
```

B.
```
if (i==j) f = g-h;
f = g+h;
```

C.
```
if (i!=j) f = g+h;
else      f = g-h;
```

D.
```
if (i!=j) f = g-h;
else      f = g+h;
```

E. Both (A) and (D).

F. Both (B) and (C).

Part B: Multiple-Response Questions [Total: 5×3=15 marks]

Each multiple-response question (MRQ) is worth **THREE marks** and may have one answer or multiple answers. Write out **all** correct answers. For example, if you think that A, B, C are the correct answers, write A, B, C.

Only if you get all the answers correct will you be awarded three marks. **No partial credit will be given for partially correct answers.**

You may write your answers into the boxes in the Answer Sheets provided or if you are using your own paper, write your answers on a **single line** to conserve space. For example:

7. A,B 8. B,D 9. C 10. A,B,C,D 11. B,D,C

Please write in **CAPITAL LETTERS**.

7. On a particular 32-bit machine with byte-addressable memory, we have the following four bytes in memory (the starting address 0x3E8 is divisible by 4):

Address	Value
0x3E8	0x43
0x3E9	0x61
0x3EA	0x74
0x3EB	0x73

Which of the following statements are true?

A. If the system is big endian, the four bytes form the word “Cats”.
B. If the system is little endian, the four bytes form the word “Cats”.
C. If the system is big endian, the four bytes form the integer value 1130460273 in decimal.
D. If the system is little endian, the four bytes form the integer value 1937006915 in decimal.

8. Dueet has written the following piece of MIPS assembly code. The first instruction (addi $3, $zero, 0) is at memory location 0x0.

```
       addi $3, $zero, 0      # instruction 1
       addi $5, $zero, 0      #
loop:  addi $3, $3, 1         # instruction 3
       andi $4, $3, 1         # instruction 4
       bne  $4, $zero, loop   # instruction 5
       addi $5, $5, 1         #
       j    loop              # instruction 7
```

Since Dueet does not have an assembler, he is assembling this code by hand. Unfortunately, he has made some mistakes. Select the instructions that were **wrongly** assembled.

A. 0x20030000 for instruction 1

B. 0x20630001 for instruction 3

C. 0x30830001 for instruction 4

D. 0x1480FFFE for instruction 5

E. 0x08000002 for instruction 7

9. Which of the following statements are **true** about number systems?

A. In an *n*-bit excess *E* signed number system where $0 < E < 2^n$, the integers that can be represented range from $-E$ to $(2^n - E - 1)$.

B. In an *n*-digit diminished radix complement number system in base R, the integers that can be represented range from $-(R^n - 1)$ to $(R^n - 1)$.

C. Sign-extension works with all signed number systems.

D. Ignoring any special bit patterns, the smallest positive number that can be represented in a 32-bit IEEE-754 format is 1×2^{-127}.

E. The decimal number 23 is represented by 10111 in BCD (Binary Coded Decimal) code.

10. Choose the statements below that are **true** about the MIPS processor.

A. The processor sign-extends constants for bitwise operations.

B. Using two j statements, it is possible to jump forward to more than 2^{26} instructions away.

C. When branching forward, the displacement is always calculated relative to the instruction after the branch, but when branching backwards, the displacement is always calculated relative to the branch itself.

D. We can use a 6-to-64 decoder to decode the opcode.

E. The processor instruction format allows for up to 127 different instructions.

11. The code below adds up a series of integers in an array whose base address is in $12 and length in $13. There are three instructions missing, marked by <instr1>, <instr2> and <instr3>. The array is assumed to have at least one element.

```
        addi $2, $zero, 0
        addi $3, $zero, 0
 loop:  sll  $4, $3, 2
        add  $5, $4, $12
        lw   $5, 0($5)
        add  $2, $2, $5
        addi $3, $3, 1
        <instr1>
        <instr2>
        <instr3>
 exit:
```

Choose all instruction groups that fit into <instr1>, <instr2> and <instr3>. Some groups may not have an <instr3> and this is indicated in the comment for that group.

A.
```
slt $5, $13, $3      # <instr1>
beq $5, $zero, loop  # <instr2>
                     # <instr3> – Nothing
```

B.
```
slt $5, $3, $13      # <instr1>
bne $5, $zero, loop  # <instr2>
                     # <instr3> – Nothing
```

C.
```
slt $5, $3, $13      # <instr1>
bne $5, $zero, exit  # <instr2>
j       loop         # <instr3>
```

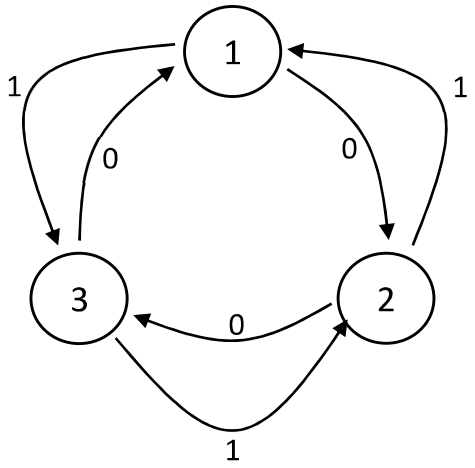
D.
```
slt $5, $13, $3      # <instr1>
bne $5, $zero, exit  # <instr2>
j       loop         # <instr3>
```

E.
```
slt $5, $3, $13      # <instr1>
beq $5, $zero, exit  # <instr2>
j       loop         # <instr3>
```

Part C: There are 5 questions in this part [Total: 70 marks]

Q12. Sequential circuits [12 marks]

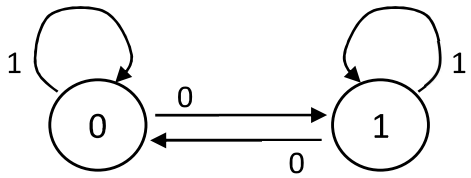
(a) Given the following state diagram on 3 states with external input *x*, where the states are shown in decimal, design the sequential circuit using two *JK* flip-flops called *A* and *B*. You do not need to produce the logic diagram of your circuit.



(i) Write out the flip-flop input functions for flip-flops *A* and *B* in SOP expressions. [4 marks]

(ii) Complete the state diagram on the Answer Sheets by drawing the transitions from unused state 0. [2 marks]

(b) Polfpilf wants to design his own *F* flip-flop with an external input *F*. The state diagram of his flip-flop is shown below.



(i) Design this *F* flip-flop using a *T* flip-flop and one logic gate. [3 marks]

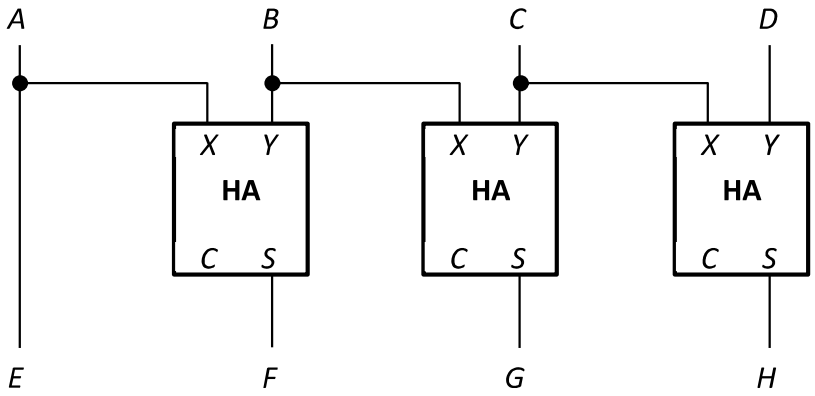
(ii) Design this *F* flip-flop using a *D* flip-flop and one logic gate. [3 marks]

Q13. Combinational circuits [13 marks]

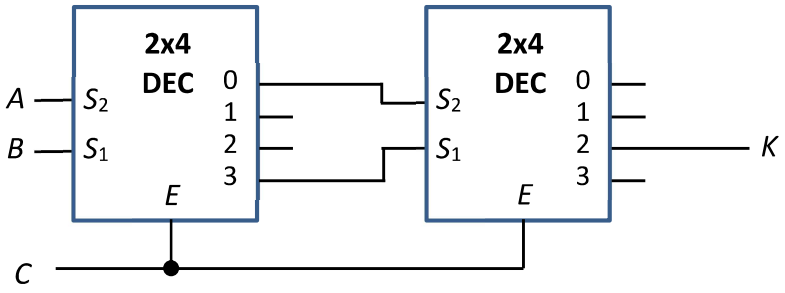
Note that logical constants 0 and 1 are available, but not complemented literals.

(a) The circuit below shows 3 half-adders. Each half adder takes inputs *X* and *Y* and produces outputs *C* (carry) and *S* (sum). The circuit takes in a 4-bit input *ABCD* and generates a 4-bit output *EFGH*.

Write out the Σm notation for the functions *E*(*A*,*B*,*C*,*D*), *F*(*A*,*B*,*C*,*D*), *G*(*A*,*B*,*C*,*D*), and *H*(*A*,*B*,*C*,*D*). [4 marks]



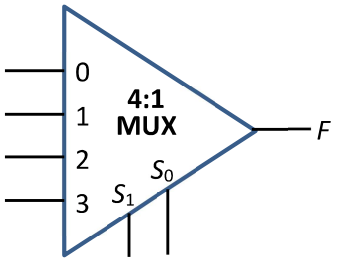
(b) The circuit below shows two 2×4 decoders with 1-enable and active high outputs. What is the simplified SOP expression for *K*? [4 marks]



(c) Given the following Boolean function

$$F(A,B,C,D) = \Sigma m(7, 10, 14) + \Sigma d(2,6),$$

implement *F* using a single 4:1 multiplexer as shown on the right with <u>at most one logic gate</u>. Once you have used a variable on a selector line, you should not use the same variable on the multiplexer inputs. [5]



Q14. MIPS [13 marks]

Study the following MIPS code on integer arrays *A* and *B* which contain the same number of elements. The following are the variable mappings:

- $a0 = *size* (number of elements in array *A*)
- $a1 = base address of array *A*
- $a2 = base address of array *B*
- $t9 = *answer*

```
.data
A: .word 10, 21, 12, 17, 9,  1, 20, 33
B: .word  7,  3, 20, 15, 2, 18, 35, 11
size: .word 8

.text
main: la   $t0, size      # $t0 is the address of size
      lw   $a0, 0($t0)    # $a0 is the content of size
      la   $a1, A         # $a1 is the base address of array A
      la   $a2, B         # $a2 is the base address of array B
# ------------------------------------------------------------
      add  $t0, $0, $0    # I1; i = 0
      add  $t9, $0, $0    # I2; ans = 0
      addi $t1, $a1, 0    # I3; $t1 = &A[0]
      addi $t2, $a2, 0    # I4; $t2 = &B[0]
      sll  $t8, $a0, 2    # I5; $t8 = size * 4
loop: slt  $t3, $t0, $t8  # I6; i < size * 4?
      beq  $t3, $0, end   # I7
      lw   $s1, 0($t1)    # I8; $s1 = A[i]
      lw   $s2, 0($t2)    # I9; $s2 = B[i]
      slt  $t3, $s1, $s2  # I10
      bne  $t3, $0, else  # I11
      add  $t9, $t9, $s1  # I12; ans = ans + A[i]
      j    cont           # I13
else: sub  $t9, $t9, $s2  # I14; ans = ans - B[i]
cont: addi $t0, $t0, 4    # I15
      addi $t1, $t1, 4    # I16
      addi $t2, $t2, 4    # I17
      j    loop           # I18
# ------------------------------------------------------------
end:  li   $v0, 1         # system call code for print_int
      add  $a0, $t9, $0   # transfer $t9 to $a0 for printing
      syscall             # print $s5
      li   $v0, 10        # system call code for exit
      syscall
```

Q14. (continue…)

(a) What output does the above MIPS code produce? [2]

(b) Using the variable names (*A*, *B*, *size*, *answer*) shown in the variable mappings above, write an equivalent C code corresponding to instructions I1 to I18 in the above MIPS code. You may use additional variable(s) if needed.

Do not do a line-by-line direct translation of the MIPS code. You do not need to declare the variables in your C code. [4]

(c) Write the instruction encoding of instruction I7 (`beq $t3, $0, end`) in hexadecimal. [2]

(d) Write the instruction encoding of instruction I8 (`lw $s1, 0($t1)`) in hexadecimal. [2]

(e) Write the instruction encoding of instruction I18 (`j loop`) in hexadecimal, assuming that instruction I1 (`add $t0, $0, $0`) is stored at address **0x2B00 0FFC**. [3]

Q15. Pipelining [14 marks]

Refer to the following MIPS code which is the same as the one in question 14. Here, we look only at instructions I1 to I18.

```
      add  $t0, $0, $0    # I1
      add  $t9, $0, $0    # I2
      addi $t1, $a1, 0    # I3
      addi $t2, $a2, 0    # I4
      sll  $t8, $a0, 2    # I5
loop: slt  $t3, $t0, $t8  # I6
      beq  $t3, $0, end   # I7
      lw   $s1, 0($t1)    # I8
      lw   $s2, 0($t2)    # I9
      slt  $t3, $s1, $s2  # I10
      bne  $t3, $0, else  # I11
      add  $t9, $t9, $s1  # I12
      j    cont           # I13
else: sub  $t9, $t9, $s2  # I14
cont: addi $t0, $t0, 4    # I15
      addi $t1, $t1, 4    # I16
      addi $t2, $t2, 4    # I17
      j    loop           # I18
end:
```

Assuming a 5-stage MIPS pipeline and *A*[0] is larger than *B*[0], answer the parts below. You need to count until the last stage of instruction I18.

(a) How many cycles does this code segment take to complete its execution in the first iteration (I1 to I18) in an ideal pipeline, that is, one with no delays? [2]

For parts (b) to (d) below, given the assumption for each part, how many additional cycles does this code segment (I1 to I18) take to complete its execution in the first iteration as compared to an ideal pipeline computed in (a)? Note that the jump instruction (j) computes the target address to jump to in its ID stage (stage 2). No delayed branching is used.

Write the total number of additional delay cycles for each of the parts (b) to (d). For example, if part (b) takes 30 cycles and part (a) takes 10 cycles, then you should write 20 for part (b).

(b) Assuming without forwarding and branch decision is made at MEM stage (stage 4). No branch prediction is made. [3]

(c) Assuming with forwarding and branch decision is made at MEM stage (stage 4). No branch prediction is made. [3]

(d) Assuming with forwarding and branch decision is made at ID stage (stage 2). Branch is predicted not taken. [3]

(e) Assuming the setting in part (b) above (without forwarding and branch decision at MEM stage), without affecting the correctness of the code, is it possible to move one instruction to somewhere else to reduce the number of delay cycles? If so, indicate which instruction to move, where to move it to, and how many delay cycles are reduced by moving it. If it is not possible, explain. [3]

Q16. Cache [18 marks]

Refer to the following MIPS code which is the same as the one in question 14. Here, we look only at instructions I1 to I18. The data segment in the MIPS code in question 14 no longer applies here as the arrays contain a lot more elements in this question.

```
      add  $t0, $0, $0    # I1
      add  $t9, $0, $0    # I2
      addi $t1, $a1, 0    # I3
      addi $t2, $a2, 0    # I4
      sll  $t8, $a0, 2    # I5
loop: slt  $t3, $t0, $t8  # I6
      beq  $t3, $0, end   # I7
      lw   $s1, 0($t1)    # I8
      lw   $s2, 0($t2)    # I9
      slt  $t3, $s1, $s2  # I10
      bne  $t3, $0, else  # I11
      add  $t9, $t9, $s1  # I12
      j    cont           # I13
else: sub  $t9, $t9, $s2  # I14
cont: addi $t0, $t0, 4    # I15
      addi $t1, $t1, 4    # I16
      addi $t2, $t2, 4    # I17
      j    loop           # I18
end:
```

For parts (a) to (d): You are given a **direct-mapped data cache** with 512 words in total and each block contains 8 words. You may assume the following:

- Array *A* starts at address **0xCDEF 0400**.
- Array *B* follows immediately after array *A* in the memory. That is, if the last element of array *A* is at address *x*, then the first element of array *B* is at address (*x* + 4).

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

(b) Assuming that array *A* contains **1032** elements and array *B* contains the same number of elements, what is the hit rate of the data cache (i) for array *A* and (ii) for array *B*? Write your answers in fractions if they are not equal to zero. [2]

(c) Assuming that array *A* contains **1028** elements and array *B* contains the same number of elements, what is the hit rate of the data cache (i) for array *A* and (ii) for array *B*? Write your answers in fractions if they are not equal to zero. [4]

(d) Assuming that arrays *A* and *B* are non empty, (i) what is the lowest hit rate possible for array *A*? Write your answer in fraction if it is not zero; (ii) how many elements in array *A* would result in this lowest hit rate? [2]

Q16. Cache (continue…)

For parts (e) and (f): You are given a **2-way set-associative instruction cache** with **LRU** replacement policy. You may assume the following:

- There are **100 elements** in array *A* and the same number of elements in array *B*.
- All elements in array *A* are positive integers and all elements in array *B* are negative integers.
- Instruction I1 is stored at address **0x2B00 0FFC**.
- Consider only instructions I1 to I18 in the execution of the code. (Make sure the execution processes all the elements in the arrays.)

(e) The instruction cache contains 16 words in total and each block contains 4 words.

(i) How many bits are there in the set index field? In the byte offset field? [2]

(ii) How many misses are there in the execution of the code? [3]

(f) The instruction cache contains 16 words in total and each block contains 2 words.
How many misses are there in the execution of the code? [3]

=== END OF PAPER ===