The University of Nottingham Ningbo China

SCHOOL OF COMPUTER SCIENCE

A LEVEL 1 MODULE, AUTUMN SEMESTER, 2022–2023

COMPUTER FUNDAMENTALS

Time allowed 60 Minutes

Candidates may complete the front covers of their answer books and sign their desk cards but must NOT write anything else until the start of the examination period is announced.

Answer all questions. The total mark is 50.

Only silent, self contained calculators with a Single-Line Display or Dual-Line Display are permitted in this examination.

Dictionaries are not allowed with one exception. Those whose first language is not English may use a standard translation dictionary to translate between that language and English provided that neither language is the subject of this examination. Subject specific translation dictionaries are not permitted.

No electronic devices capable of storing and retrieving text, including electronic dictionaries, may be used.

DO NOT turn examination paper over until instructed to do so

Question 1 (25 marks)

(a) Convert the following 8-bit two's complement binary into decimal.
$a = 1011\ 0011_2$ $b = 0110\ 0011_2$ $c = 1111\ 0000_2$ (3 marks)

(b) Convert the following decimal values to 8-bit two's complement binary.
$a = -123$ $b = -79$ $c = 41$ (3 marks)

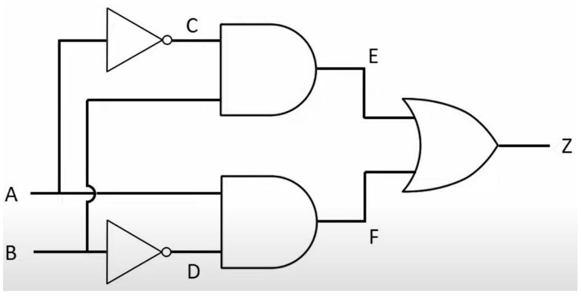
(c) Simplify the following Boolean Expression.

(i) $C(CB + \overline{A}B) + \overline{BA}$

(ii) $(CD + \overline{C}B)C + \overline{B}B$

(iii) $C(BD + B(BD)) + \overline{CC}$ (6 marks)

(d) Complete the truth table for the below composite gate. (8 marks)



A	B	C	D	E	F	Z
0	0					
0	1					
1	0					
1	1					

(e) Write the pseudocode to illustrate fetch-execute cycle in the processor. (5 marks)

END OF QUESTION 1

Question 2 (25 marks)

(a) This question is based on the symbolic assembly code below:

```
      @2
      D=A
      @i
      M=D
      @s
      M=0
   (LOOP)
      @i
      D=M
      @3
      D=D-A
      @END
      D;JGT
      @i
      D=M
      @s
      M=D+M
      @i
      M=M+1
      @LOOP
      0;JMP
   (END)
      @END
      0;JMP
```

(i) Derive the value of register **RAM[17]** after the execution of this piece of code. (4 marks)

(ii) Convert the last two lines of the symbolic assembly code in (a),

```
@END
0;JMP
```

to binary machine code. You may refer to APPENDIX 1 and 2 for this conversion. (2 marks)

(b) In general, from **high-level programming language code** all the way down to the computer hardware, we have **virtual machine code**, **symbolic assembly code** and **binary machine code** in between.

Draw the block diagram for those building blocks, and link them properly with the suitable **programming language translators**. (3 marks)

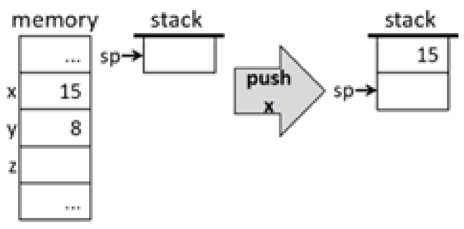
(c) This question is based on the VM code below:

```
push constant 0
pop local 0
label LOOP_START
push argument 0
push local 0
add
pop local 0
push argument 1
push constant 1
sub
pop argument 1
push argument 1
if-goto LOOP_START
push local 0
```

(i) Show the final stack status. Assume initially the memory location **argument**[**0**] takes the value of 3 and the memory location **argument**[**1**] takes the value of 4. (4 marks)

(ii) What operation is performed by the VM code in (c)? Where is the memory location of the final result? (2 marks)

(d) Show the stack operations and the final memory status for the following operations: $\boldsymbol{z = (x > 7) and (x + y < 30)}$
The initial memory status and the first stack operation are shown below. (5 marks)



(e) Translate the VM command **"push local 5"** into **Hack symbolic assembly code**. (5 marks)

END OF QUESTION 2

APPENDIX 1 A-instruction specification

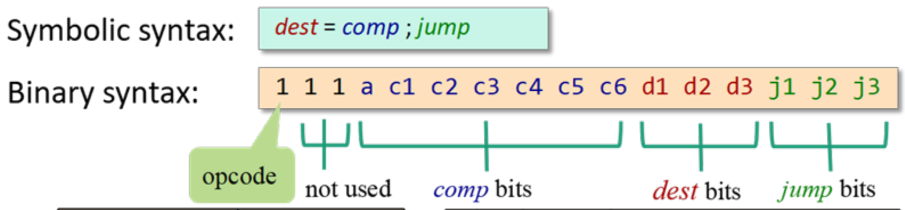
Symbolic syntax:

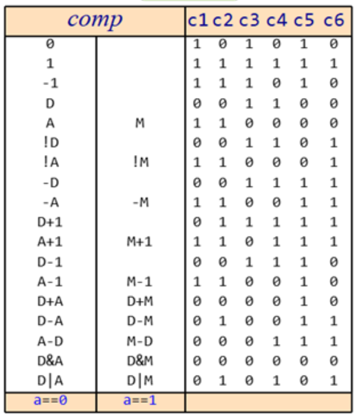
@value

Binary syntax:

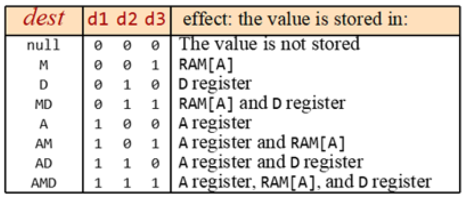
0value

APPENDIX 2 C-instruction specification

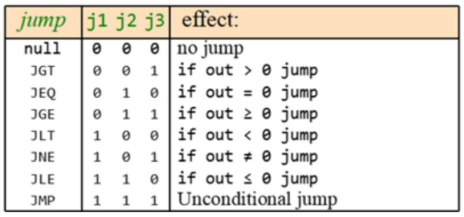




comp		c1	c2	c3	c4	c5	c6
0		1	0	1	0	1	0
1		1	1	1	1	1	1
-1		1	1	1	0	1	0
D		0	0	1	1	0	0
A	M	1	1	0	0	0	0
!D		0	0	1	1	0	1
!A	!M	1	1	0	0	0	1
-D		0	0	1	1	1	1
-A	-M	1	1	0	0	1	1
D+1		0	1	1	1	1	1
A+1	M+1	1	1	0	1	1	1
D-1		0	0	1	1	1	0
A-1	M-1	1	1	0	0	1	0
D+A	D+M	0	0	0	0	1	0
D-A	D-M	0	1	0	0	1	1
A-D	M-D	0	0	0	1	1	1
D&A	D&M	0	0	0	0	0	0
D|A	D|M	0	1	0	1	0	1
a==0	a==1						



dest	d1	d2	d3	effect: the value is stored in:
null	0	0	0	The value is not stored
M	0	0	1	RAM[A]
D	0	1	0	D register
MD	0	1	1	RAM[A] and D register
A	1	0	0	A register
AM	1	0	1	A register and RAM[A]
AD	1	1	0	A register and D register
AMD	1	1	1	A register, RAM[A], and D register



jump	j1	j2	j3	effect:
null	0	0	0	no jump
JGT	0	0	1	if out > 0 jump
JEQ	0	1	0	if out = 0 jump
JGE	0	1	1	if out ≥ 0 jump
JLT	1	0	0	if out < 0 jump
JNE	1	0	1	if out ≠ 0 jump
JLE	1	1	0	if out ≤ 0 jump
JMP	1	1	1	Unconditional jump