The University of Nottingham

SCHOOL OF COMPUTER SCIENCE

A LEVEL 1 MODULE, AUTUMN SEMESTER 2019-2020

COMPUTER FUNDAMENTALS (COMP1036)

Time allowed: **60 Mins**

Candidates may complete the front cover of their answer book and sign their desk card but must NOT write anything else until the start of the examination period is announced

Answer All Questions

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 your examination paper over until instructed to do so

Question 1

a. State formulae for De Morgan's Law and the Distributive Rule [4 marks]

b. Draw the truth table for a 1 bit full adder [3 marks]

c. Convert the binary number 11011001 to an 8 bit, 2s complement decimal number [2 marks]

d. Using only NAND gates draw a gate diagram for XOR [4 marks]

e. Derive the minimal Boolean expression using Boolean algebraic techniques and draw the gate diagram for the canonical representation of the following truth table

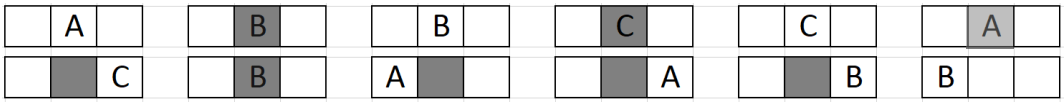
[8 marks]

X	Y	Z	OUT
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

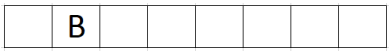
f. Discuss how different types of delays effect the performance of a ripple carry adder.

(4 marks)

g. Consider a generalization of a Turing machine, in which the head is allowed to move in either direction is considered. 6 relevant rules of the 3-state (A, B, C), 2-colour((dark ■ light □), 3-direction (left, right, static) Turing machine are shown below.



Assuming dark = 1 and light = 0 and the starting state of an 8 bit word is:



What will the final 8-bit binary number be once the machine reaches a stable state? (5 marks)

Question 2

a. This question is about machine language. Based on the symbolic assembly code below,

```
  @5
  D=A
  @R0
  M=D

  @R0
  D=M
  @n
  M=D
  @pre
  M=0
  @cur
  M=1

(LOOP)
  @cur
  D=M
  @n
  D=D-M
  @STOP
  D;JGT

  @pre
  D=M
  @cur
  D=D+M
  @nex
  M=D

  @cur
  D=M
  @pre
  M=D

  @nex
  D=M
  @cur
  M=D
  @LOOP
  0;JMP

(STOP)
  @nex
  D=M
  @R1
  M=D

(END)
  @END
  0;JMP
```

I. Please derive the value of RAM[1] after the execution of this piece of code. [4 Marks]

II. Please convert the first two lines of the symbolic assembly code in (a),

```
@5
D=A
```

to **binary machine code**. You may refer APPENDIX 2 for this conversion. [2 Marks]

III. Please convert the last two lines of the symbolic assembly code in (a),

```
@END
0;JMP
```

to **binary machine code**. You may refer APPENDIX 2 for this conversion. [2 Marks]

b. This question is about machine language. Please implement the following in **symbolic assembly language**. SUM is a variable. You **MUST** use built-in symbols. The built-in symbols are given in APPENDIX 3. You **MUST** terminate the program properly. You **MUST** use label (END).

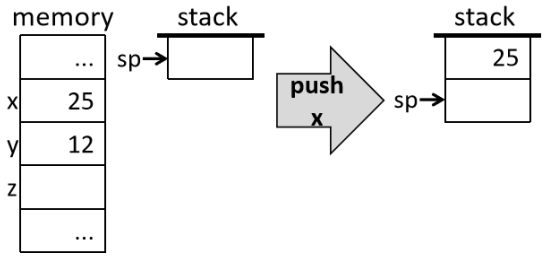
```
RAM[0] = 10;
RAM[1] = 20;
SUM = RAM[0]+RAM[1];
```

[4 Marks]

c. This question is about stack operation. Please derive the **stack operations**, the **stack status** for each operation, and the **final memory status** for the following operations:

z = (x-y>10) and (x<15).

The initial memory status and the first operation are shown below.



[5 Marks]

d. This question is about stack implementation. Translate the VM command "**push constant 10**" into **hack symbolic assembly code**. (**Hint**: the corresponding hack pseudo-code is: ***SP = 10; SP++**). [3 Marks]

End of Question 2: Total 20 marks

APPENDIX

1. A-instruction specification

Symbolic syntax:

@value

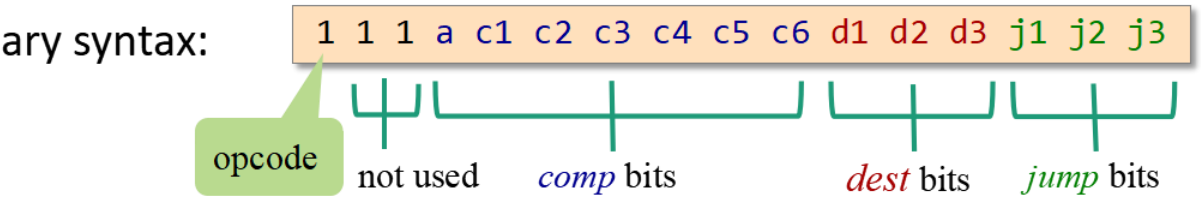
Binary syntax:

0value

2. C-instruction specification

Symbolic syntax: *dest* = *comp* ; *jump*

Binary syntax:



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

3. Build-in symbols of Hack assembly code.

symbol	value
R0	0
R1	1
...	...
R15	15
SCREEN	16384
KBD	24576

symbol	value
SP	0
LCL	1
ARG	2
THIS	3
THAT	4