The University of Nottingham

SCHOOL OF COMPUTER SCIENCE

A LEVEL 1 MODULE, AUTUMN SEMESTER 2018-2019

COMPUTER FUNDEMENTALS

Time allowed: **One Hour**

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 (30 Marks)

a. State De Morgans Law

[2 marks]

b. Draw the gate diagram for the canonical representation of the following truth table

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

[5 marks]

c. Using only Nand gates give the Boolean expressions for And, Not and Or

[3 marks]

d. Briefly explain the '2s complement' method of representing negative numbers and the 'Sign and Magnitude' approach, discuss the merits of each

[4 marks]

e. Discuss how different types of delays within a chip effects the maximum possible clock speeds

[4 marks]

f. Compare, with gate diagrams, the full adder with the half adder

[4 marks]

g. Implement a circuit for the following function: If $sel_1 = 1$ then out = a;

else if $sel_2 = 1$ then out = b, else out = c

[4 marks]

h. Briefly state how the Universal Turning Machine relates to the Von Neumann Architecture and draw an example of a Von Neumann structure

[4 marks]

Question 2 (20 Marks)

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

```
@4
D=A
@R0
M=D

@R0
  D=M
  @n
  M=D
  @i
  M=0
  @sum
  M=0

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

@sum
  D=M
  @i
  D=D+M
  @sum
  M=D
  @i
  M=M+1
  @LOOP
  0;JMP

(STOP)
  @sum
  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 (i),

```
@4
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 (i),

```
@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. You should use build-in symbols. The build-in symbols are given in APPENDIX 3. You should terminate the program properly. You may use label (END).

RAM[0] = 10;

RAM[1] = 20;

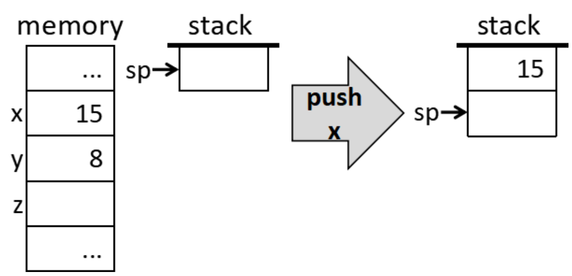
RAM[2] = RAM[0] + RAM[1];

[4 Marks]

(c) In general, from high-level programming language all the way down to the computer hardware, we have virtual machine, symbolic assembly code and binary machine code in between. Please draw the block diagram for those building blocks, and **link them properly**. E.g. from high-level programming language to virtual machine, we need a compiler.

[3 Marks]

(d) This question is about stack operation. Please derive the stack operation and final memory status for the following operations: z = (x>7) and (x+y>30). The initial memory status and the first operation is shown below.



[5 Marks]

APPENDIX

1. A-instruction specification

Symbolic syntax:

@value

Binary syntax:

0value

2. C-instruction specification

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

Binary syntax: 1 1 1 a c1 c2 c3 c4 c5 c6 d1 d2 d3 j1 j2 j3

opcode | not used | *comp* bits | *dest* bits | *jump* bits

comp (a==0)	*comp* (a==1)	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