IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2015

EEE PART II: MEng, BEng and ACGI

INTRODUCTION TO COMPUTER ARCHITECTURE

Friday, 12 June 10:00 am

Time allowed: 1:30 hours

Corrected Copy

There are THREE questions on this paper.

Answer ALL questions.

All questions carry equal marks

Any special instructions for invigilators and information for candidates are on page 1.

Examiners responsible	First Marker(s) :	T.J.W. Clarke
	Second Marker(s) :	C. Bouganis

The Questions

1. a) Convert the 32 bit IEEE-754 format number 0x3E300000 to decimal, showing your method.

[4]

b) A 32 bit ARM CPU has a direct mapped cache of size 1024 lines each of 32 words.

i) What is the total data size of the cache not including tag storage. Give your answer in bytes.

[1]

ii) Which ARM address bit fields constitute the tag, select and index fields of this cache?

[3]

iii) State, giving your working, what is the total number of bits required to store tag values in the cache?

[2]

iv) The CPU performs a sequence of contiguous word READ operations such that the *n*th operation has address 4**n*. What is the ratio of misses to hits if this sequence continues for a long time? Is your answer affected by the number of lines in the cache?

[4]

c) Write a fragment of ARM code which sets R0 equal to the least significant 32 bits of 16*R1 + 16385*R2. All values are unsigned. R1 and R2 must not be changed. Credit will be given for efficient implementations that use the minimum number of registers. You may find it useful to note that 16384 is a power of 2.

[4]

d) State, giving reasons, whether your code in part c) would change given the same requirement on *two's complement signed* values of R0, R1 and R2 given that R0 must still equal the least significant 32 bits of the answer.

[2]

2. a) A fragment of assembly code containing loops is shown in Figure 2.1.

i) For each branch in the code state whether it is escape, looping, or skipping.

[4]

ii) State the number of times that LOOP1 executes.

[1]

iii) For a single execution of LOOP1, state the number of times that LOOP2 executes and LOOP3 executes.

[2]

iv) When this code fragement is executed from START through to FINISH the *execution count* of each instruction is defined to be the number of times it is condition true executed. State the largest execution count of any instruction in this code fragment, and the set of instructions that have this maximum execution count.

[3]

4* 4* —— 10:40

b) This part concerns an ARM block copy subroutine which has inputs R0, R1, R2. The subroutine copies R2 words of memory:

$$\text{mem}_{32}[\text{R0}+i] \rightarrow \text{mem}_{32}[\text{R1}+i] \quad \text{where } 0 \leq i \leq \text{R2}-1$$

You may assume that the source and destination blocks of memory do not overlap and that both R0 and R1 are word-aligned pointers. You may assume that R2 is non-zero and positive.

i) Write an ARM code fragment that performs this block move, with inputs R0,R1,R2 as specified. Credit will be given for code which has a small number of instructions. Load/Store Multiple (LDM/STM) instructions may not be used.

[4]

ii) Suppose the source and destination blocks of memory overlap as shown in Figure 2.2. Explain in words how a block copy loop would need to be written to work in this case.

[2]

iii) Write the subroutine entry and exit code such that no register value is changed by the subroutine, assuming a stack which moves upwards on PUSH and has stack pointer R13 pointing to an empty location.

[4]

in address

11:00

```
START   MOV   R2, #4
LOOP1   MOV   R1, #100
A       MOV   R0, R1
LOOP2   SUBS  R0, R0, #1
B       BNE   LOOP2
C       MOV   R0, R1
LOOP3   CMP   R0, #0
D       SUB   R0, R0, #1
E       BNE   LOOP3
F       SUBS  R2, #1          10:30
G       BMI   FINISH
H       B     LOOP1
FINISH
```

Figure 2.1: Assembler Loop.

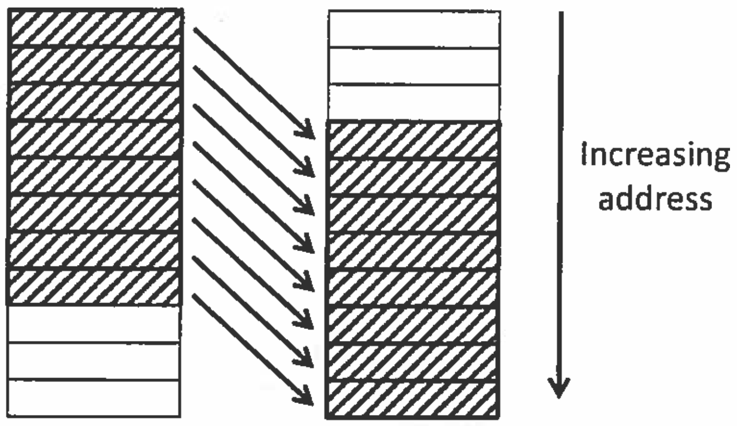

Figure 2.2: Overlapping source and destination blocks.

3. The code shown in Figure 3.1 has input R1 and output R2. You may assume that R1 has precisely 1 non-zero bit and lies in the range $1 \leq \text{R1} \leq 2^7$. The label TABLE points to a table of constant data as shown in Figure 3.1, where each number in the DCB statement determines the value of one 8 bit byte. The code outputs in R2 the number of the bit set to 1 in the binary repreprestation of R1. Throughout this question the least significant bit is numbered 0.

a) Showing all your working, state how many cycles are taken by this code. Explain why it is advantageous to use LDRNEB rather than LDRB for the instruction with label S1.

[4]

b) If the value in R1 is written as $16a+b$, where $0 \leq a,b \leq 15$ and as before R1 has 1 non-zero bit, determine the memory addresses used in the instructions with labels S1 and S2 as functions of a, b and TABLE. Show your reasoning.

[4]

c) Explain how the algorithm works and state which of the TABLE data locations, if any, have values on which the algorithm correctness does *not* depend.

[4]

d) Write a new version of the code in Figure 3.1 using a different table of length 16 bytes and modified code which returns in R2 a number equal to the number of bits set to 1 in the binary representation of R1, where R1 is known to satisfy $0 \leq \text{R1} \leq 255$. Explain clearly your method.

[8]

```
        ADR      R10, TABLE
        ANDS     R0, R1, #0xF
S1      LDRNEB   R2, [R10, R0]
        ANDS     R0, R1, #0xF0
S2      LDRNEB   R2, [R10, R0, LSR #4]
        ADDNE    R2, #4

TABLE   DCB      0, 0, 1, 0, 2, 0, 0, 0, 3
```

Figure 3.1: Code.