IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2014

EEE PART II: MEng, BEng and ACGI

Corrected Copy

INTRODUCTION TO COMPUTER ARCHITECTURE

Friday, 13 June 10:00 am

Time allowed: 1:30 hours

There are THREE questions on this paper.

Answer ALL questions.

Question 1 carries 40% of the marks. Questions 2 and 3 carry equal marks (30% each).

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) : Y.K. Demiris

The Questions

1. a) A new CPU architecture, the ARM X, has a 10 stage pipeline and stalls of length 8 machine cycles. The performance of this for different classes of branch instructions, together with the probability with which each class is executed, is shown in Figure 1.1. You may assume that the ARM X pipeline stalls occur only as the result of branches. If the ARM X CPU machine cycle frequency is 1.2GHz determine, showing your working, what is the typical throughput of the ARM X in millions of instructions per second (MIPS).

[5]

b) Convert the 32 bit IEEE-754 format number 0xC008,0000 to decimal, showing your method.

[4]

c) Write a fragment of ARM code which implements the 64 bit addition:

R3:R2 := R1:R0 + 0x2400,0000,0001

Each pair of registers Ra:Rb represents a 64 bit unsigned number with Ra, Rb respectively the most, least significant 32 bits. The answer in R3:R2 must contain the least significant 64 bits of the result. Credit will be given for time-efficient solutions.

[5]

d) Write a fragment of ARM code which copies memory with byte addresses R10, ..., R10+15 to memory with byte addresses R11, ..., R11+15, preserving the order of the bytes in memory. You may assume that these two contiguous memory areas do not overlap and that both R10 and R11 are word-aligned (divisible by 4). Your code may change any of the registers R0-R9 but must not change R10, R11. Credit will be given for time efficient solutions.

[6]

Branch type	Branch execution probability	Incorrect prediction probability
Unconditional	0.1	0
Conditional	0.2	0.25
Computed	0.05	1

Figure 1.1: Branch Prediction.

2. a) The code fragment in Figure 2.1 executes with all condition codes and registers initially 0. The label DATA is assigned value 0x100 by the assembler. State the values indicated in Figure 2.2, when execution of the code fragment reaches label FINISH. Write your answers in the form indicated by Figure 2.2. Your answer must be written with register values in signed decimal (55, -31), memory location values in hexadecimal, and condition codes in binary. Note the order required for the condition codes.

example

[10]

b) An ARM bit processor uses a direct-mapped write-through cache which has 8 lines each of 128 bytes. Initially all lines in the cache are invalid (Valid=0). The sequence of word CPU memory operations shown in Figure 2.3 is executed by the CPU in the specified order. For each CPU operation, state whether it is a HIT or a MISS, precisely what (possibly null) set of main memory operations is required to implement the CPU operation using the cache, and what is the tag field of the affected cache line after the operation.

[10]

```
        MOV   R0, #3
        ADR   R10, DATA
        ADR   R11, DATA1
        SUB   R2, R11, R10
LOOP    LDR   R1, [R10, R0, LSL #2]
        STRB  R1, [R11, R0]
        SUBS  R0, R0, #1
        BGE   LOOP
FINISH  JMP   FINISH
DATA    DCD   0x12345678, 0x11223344
DATA1   DCD   0x10203040, 0x00000000
```

Figure 2.1: Code fragment.

R0	R1	R2	R10	CV NZ	Memory bytes (byte addresses)			
					[0x103]	[0x102]	[0x101]	[0x100]

Figure 2.2: Template for answers.

order	operation	address
1	READ	0x400
2	READ	0x900
3	READ	0x410
4	WRITE	0x970
5	WRITE	0x000

Figure 2.3: CPU memory operations.

3. a) Write a single fragment of ARM code using the minimum possible number of ARM instructions that will implement multiplication and division as specified in Figure 3.1. All results are truncated to 32 bits.

[6]

b) The ARM fragment shown in Figure 3.2 has input R0 and outputs R1, R2. Using the ARM timings from the exam notes, state the execution time in machine cycles, and the values in R1 and R2 when the code reaches label FINISH, for execution of this code fragment with R0 initially equal to 0x2300,0000.

[3]

c) The fragment in Figure 3.2 is to be used as a subroutine in which R2 is the only output, R1 is not required. There is an ascending (on PUSH) stack where the stack pointer R13 points to the lowest full stack location. Add optimised code before and/or after this fragment to turn it into a subroutine such that no register is changed other than output.

[3]

d) The code in Figure 3.2 is partially *unrolled* by repeating the ADDPL and MOVPLS instructions N times, for some small value of N, as shown in Figure 3.3 where $N = 2$.

i) Explain why the unrolled code has identical results to that in Figure 3.2 for all values of $N \geq 1$.

[3]

ii) Determine the execution time of such unrolled code, for data where the original loop is known to execute for P iterations, as a function of P, N and $\lceil P/N \rceil$. $\lceil x \rceil$ is the smallest integer greater than or equal to x.

[3]

iii) Hence determine the value of N for minimum execution time with initial value of R0 = 0x800,0000.

[2]

Output Register	Value	Notes
R1	$2049 \times R0$	
R2	$R0/2^5$	Integer result treating R0 as *unsigned*. Your answer need be correct only for values of R0 divisible by 32.
R3	$R0/2$	Integer result treating R0 as *signed*. Your answer need be correct only for even values of R0.

Figure 3.1: Outputs from code fragment.

```
NORMALISE   MOV     R2, #0
            MOVS    R1, R0
LOOP        ADDPL   R2, R2, #1      ;Add
            MOVPLS  R1, R1, lsl #1  ;Shift
            BPL     LOOP
FINISH
```

Figure 3.2: ARM Code fragement for part b).

```
NORMALISE2  MOV     R2, #0
            MOVS    R1, R0
LOOP        ADDPL   R2, R2, #I      ;Add
            MOVPLS  R1, R1, lsl #1  ;Shift
            ADDPL   R2, R2, #1      ;Add
            MOVPLS  R1, R1, lsl #1  ;Shift
            BPL     LOOP
FINISH
```

Figure 3.3: ARM Code fragment for part d).