Introduction to Computer Architecture - Answers 2013

All questions are compulsory, questions are weighted equally.

Answer to Question 1

Qa,b,d are easy questions testing basic knowledge & understanding.C tests ability to optimise code (i) is bookwork.

1.

a)

(i) 13631489 (2^{23}*0b1.101 000 000 000 0001) **2**

(ii) 13631490 (2^{23}*0b1.101 000 000 000 0010) **2**

(iii) As (i) but exp is one larger => X2.

Note that the exp=24, so difference between successive IEEE-754 numbers in this range is 2. Half of this (1) is the max deviation from the nominal value:

27262977 – 27262979 (13631489*2 ±1) **2**

Common mistakes were to get exponent wrong, or to give an approximate answer (calculator will not have enough precision unless used carefully)

[6]

b)

(i) 0x**34D25** **1**

(ii) **0x03** **1**

(iii) **invalid, will cause memory abort exception** **1**

Many did not identify that (iii) was invalid, and that it would result in an exception.

[3]

c)

(i) ADD R1, R2, R2, lsl #6 **3**

(ii) RSC R3, R3, R4, lsl #1 ; works because C=0 **3**

Many took 2 or mor instructions to implement (ii)

[6]

d)

T = 100MHz/(1+3*0.3*0.25) = 81.6MHz (or 0.816 instructions/cycle) **3**

Typical code will contain many LDR/STR instructions which also cause pipeline stalls, so decreasing throughput. **2**

[5]

Answer to Question 2

This question tests ability to understand and analyse operation of ARM assembly code in detail. It requires accuracy and comprehensive understanding of the instructions, but is straightforward.

(a) tests understanding of two's complement arithmetic.

(b) test understanding of shift and rotate instructions.

(c) tests understanding of LDR/LDRB instructions.

Marks are awarded 1 per answer not flagged n/a. Note that condition codes CV and NZ are counted as separate answers.

Mistakes here were too varied to categorise. Worth noting that many lost marks with incorrect condition codes.

Location (word)	Value
0x100	0x04030201
0x104	0x08070605
0x108	0x0F0C0D0A
>0x108	0

Figure 2.1. Memory locations

```
MOV R4, #1
MOV R5, #-2
ADDS R0, R4, R5
SBCS R1, R5, R4
RSB R2, R4, #80000000
MVNS R3, #0
```

(a)

```
MOV   R1, #10
MOV   R2, #3
MOV   R3, #0
MOVS R2, R2, ror #1
ADDMI R3, R3, R1
MOVS R2, R2, ror #1
ADDMI R3, R3, R1, lsl #1
MOVS R2, R2, ror #1
ADDMI, R3, R3, R1, lsl #2
```

(b)

```
MOV  R4, #0x104
ANDS R3, R4, #7
LDRB  R0, [R4, #3]
LDRB  R1, [R4, #-1]
LDR   R2, [R4, R3, lsl #1]
```

(c)

Figure 2.2. Code fragments

	R0	R1	R2	R3	R4	NZ CV	Time		*Marks*
(a)	-1	-4	0x4C4B3FF 79999999	0xFFFFFFFF -1	1	10 10	n/a		*7*
(b)	n/a	10	0x60000000 1610612736	30	n/a	00 00	9		*6*
(c)	8	4	0	4	0x104 260	00 n/a	14		*7*

Figure 2.3. Template for answers

Answer to Question 3

This question tests understanding of subroutines, and operation of LDM/STM instructions, as well as how to optimise sequential access to memory.

a)

mem32[a] := mem32[b] **2**
mem32[a+4] := mem32[b+4] **1**
mem32[a+8] := mem32[b+8] **1**
execution time = 4 (STM) + 6*4 (LDR/STR) + 4(LDM) + 4 (MOV) + 4 (BL) = 40 **2**
[allow **1/2** for time if working shows STM, LDR/STR, and LDM time is all correct]

This question was well answered by those who worked out what the memory transfer was. Some gave incorrect answers in involving registers.

[6]

b)

1:
mem32[c] := R2
R13 := R13+4 (push R2)
8/9:
R2 := mem32[c]
R13 := R13 – 4 (pop R2)
PC := R14
R2 is pushed onto an empty ascending stack with SP R13, and then popped back off it at end. **2**
Finally a branch is made to the address stored in R14 (which will be the return address written by a BL instruction). **1**
R2 must be saved on stack because it is used as working register by subroutine. **1**
[4]

Many missed the reasons for the operations

c)

```
LDMIA R1, {R2,R3,R4}
```
2

```
STMIA R0, {R2,R3, R4}
```
2

24 -12 = 12 cycles **1**

Some got the direction of mem transfers wrong here.

Common mistake was to increment the base registers R1,R0, that must not happen

[5]

d)

```
1    MOVE    STMEA    R13!, {R2,R3,R4}
2            LDMIA    R1, {R2,R3,R4}
3            STMIA    R0, {R2,R3,R4}
4            LDMEA    R13!, {R2,R3,R4}
5            MOV      PC, R14
```

4

Total time saved = 12 – 4 (longer STMED/LDMED) = 8 cycles **1**

Most found this question quite easy.

[5]