THE ANSWERS

A: Analysis, B: Bookwork, D: Design

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).

A/B. Average stall time per cycle is the sum of instruction class probability times the prediction wrong probability times the stall length of 8:

$$T = \frac{1}{1+PBS} = \frac{1}{1+0.2\times 0.25\times 8+0.05\times 1\times 8} = 0.5556$$

Throughput in MIPS is 1200 * 0.5556 = 667MIPS
Correct equation 2 marks
Correct sum of probabilities 2 marks
Correct answer 1 mark *Common mistakes*

- *Using pipeline length 10 instead of stall length 8*
- *Not summing the different instruction class contributions*

[5]

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

B. -2.125
s = -1 (1 mark)
exp= 128, exponent = 1 (1 mark)
frac $= 0.0001_{(2)} = 0.0625_{(10)}$ (1 mark)
$-1\times 1.0625\times 2^1$ = -2.125 (1 mark)

[4]

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.

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.

D. The constant splits into 0x2400 upper 32 bits and 0x1 lower 32 bits. Both of these values are within the range allowed for ARM immediate constants. We ignore carry from MSW since only the low 64 bits are required. 3 marks for a correct answer of any length or 7 - n marks for a correct answer of length n cycles if this is more.

Common mistakes

- *not using immediate values in ADD instructions*
- *using explicit shift to make 0x2400 (compiler does this)*

```
ADDS    R2, R0, #1
ADC     R3, R1, #0x2400
```

[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.

D. Many solutions are possible. Word access will be faster than byte access and is possible. Multiple register transfer will be faster than separate LDR/STR. Three marks for any working solution. 4 marks for LDR/STR. 5 marks for LDR/STR loop unrolled. 6 marks for LDM/STM. Minus one mark if R10,R11 are not kept constant. Minus one mark if solution is obviously not optimal.

Common mistakes

- *not realising that 16 bytes need to be transferred. Common values: 15 bytes, 15 words, 16 words.*
- *using LDRB instaed of LDR or (better) LDM. The Q says the blocks are both word aligned so can transfer by word.*

```
LDMIA   R10, {R0-R3}
STMIA   R11, {R0-R3}
```

[6]

Question 1. has a total of 20 marks.

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.

A.

					mem$_8$			
R0	R1	R2	R10	CV NZ	[0x10B]	[0x10A]	[0x109]	[0x108]
-1	0x12345678 or 305419896	8	0x100	00 10	0x00	0x40	0x44	0x78

[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.

A. Addresses here are all words. Would accept byte block boundaries, e.g. 0x400-0x4FF. For this cache select = A6:0 and index = A9:7.

1	Miss	READ 0x400-0x47C	1
2	Miss	READ 0x900-0x97C	2
3	Hit	null	1
4	Hit	WRITE 0x970	2
5	Miss	READ 0x000-0x7C WRITE 0x000	0

Common mistakes

- *Not specifying precisely which memory locations get read or written*
- *Thinking select = A4:0*

[10]

Question 2. has a total of 20 marks.

```
        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.

					Memory bytes (byte addresses)			
R0	R1	R2	R10	CV NZ	[0x10B]	[0x10A]	[0x109]	[0x108]

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.

B.

```
ADD   R1, R0, R0, lsl #11
MOV   R2, R0, lsr #5
MOV   R3, R0, asr #1
```

Common mistakes

- *not using ASR for signed division*
- *using MUL for multiplication (slower)*
- *using MUL with immediate operand (not allowed)*

[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]

B.
Two iterations needed.
11 cycles, R1 = 0x8C00,0000, R2 = 2

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 highest 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.

C.
Here R1 needs to be stored on the stack, so we add:

```
STMFA   R13!, {R1, R14}
                        ; fragment inserted here
LDMFA   R13!, {R1, R15}
```

or could add:

```
        STMFA   R13!, {R1}
                        ; fragment inserted here
        LDMFA   R13!, {R1}
        MOV     PC, R14
```

Common mistakes

- *Saving R0 (it is never changed)*
- *Saving lots of redisters*
- *forgetting to set PC equal to R14*
- *Saving R2 (it is an 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 $N \geq 1$.

A.
The unrolled ADD/MOV instructions only execute condition true if N is 0, in which case the original version would have looped and so also execute them. When N is 0 all subsequent unrolled instructions are executed condition false and so have no effect. *Common mistakes*

- *not mentioning that N status bit controls execution, or that all instructions after N=1 or not executed.*

[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.

A.
The unrolled loop will execute for $\lceil P/N \rceil$ iterations. Each iteration will take $2N$ cycles for the iterated ADD/MOV and 4 cycles for the BPL executed condition true, $2N+4$ in total. The last iteration will be 3 cycles faster since the BPL is executed condition false. We must add the 2 initial instructions (2 cycles). So the total is: $(2N+4)\lceil P/N \rceil - 1$. *Common mistakes*

- *very few people correctly calculated the loop time as* $2N+4$
- *very few people realised that the number of loops must be* $\lceil P/N \rceil$

[3]

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

A.
0x800,000 will require 4 iterations, hence $P = 4$. Optimal $N = 4$ from inspection of equation.

[2]

Question 3. has a total of 20 marks.

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, #1       ;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).