Introduction to Microcontrollers

Class Test

12 September 2014

Student Number:

FOOBAR001

Please contain your answers to the boxes provided. They are there to indicate how much verbosity is expected. Long answers will annoy your marker.
You need to use the correct jargon to get marks.

Question 1

a) Assume you needed a programmable system to control something like an electronic sign board. Why might you prefer to use a microcontroller for this task rather than a conventional desktop computer? (2)

A microcontroller would be cheaper, smaller and consume less power.

b) Why do we load our code into Flash rather than RAM? (1)

Code will not be lost when power removed.

c) What is the difference between a microprocessor and a microcontroller? (2)

Microprocessor is only the CPU.
Microcontroller contains a CPU as well as peripherals for interfacing with various types of signals and onboard memory.

Question 2

a) The ARM Cortex-M0 is said to be *little endian*? What does this endianness model tell us about how data is laid out in memory? (1)

LSB is located at effective address.

b) Assume we wanted to place the word 0x4269AA55 at the address 0x0000 0010. Indicate which individual memory address would take on which data. (2)

Address:	0x10	0x11	0x12	0x13
Data:	0x55	0xAA	0x69	0x42

Question 3

a) How wide are each of the CPU registers? (1)

32 bits

b) Our CPU implements an instruction pipeline. List the pipeline stages, in order(1)

Fetch, Decode, Execute

c) Bearing this pipeline in mind, what is the purpose of the Program Counter? (1)

Points to instruction to be fetched.

Question 4

a) What is the main purpose of an *assembler?* Ie: what does an assembler do? (1)

Converts human readable assembly code into machine code / object code

b) Consider the appendices. What binary string does the following instruction compile to?
ADDS R1, R1, #0x0F (2)

001 10 001 00001111

c) Consider the appendices. Why will the following instruction not compile?
ADDS R7, R6, #8 (1)

Different Rd and Rn necessitates T1 encoding.
T1 encoding limits immediate data to 3 bits. #8 requires 4 bits.

d) What is meant by the terms *loading* and *storing*? (1)

Loading: Getting data from memory into CPU registers

Storing: Putting data from CPU registers into memory

e) Write down two examples of a store instruction: one which uses an immediate offset and one using a register offset. (2 x 0.5)

```
STR R0, [R0, #0]
STR R0, [R0, R0]
```

f) Assume the following block of code is executed. Which memory addresses will be loaded into R0? **List all individual byte address.**

```
MOVS R1, #0xA0
MOVS R2, #0x10
LDR R0, [R1, R2]
```

(2)

0xB0, 0xB1, 0xB2, 0xB3

g) Explain what is meant by program counter relative loading. (1)

Effective address of data to load is equal to PC value plus some offset.

h) Why are labels really useful? (1)

Gets compiler to do tedious work of calculating memory addresses of data or offsets.

i) Discuss what the assembler does when it compiles an instruction like:

```
LDR R0, foobar      ; where foobar is some label.
```

(3)

- finds address of data labeled foobar
- calculates the difference between the address of this instruction and the label
- creates a program counter relative load with that offset as the difference

j) Assuming the GPIOB_ODR contains the value 0x00000F. (2)
What value will it hold once the following block of code executes?

```
LDR R0, =0x48000400
LDR R1, [R0, #0x14]
MOVS R2, #0b10
ORRS R1, R1, R2
STR R1, [R0, #0x14]
```

0x0F

Question 5

a) What is the difference between: B{cc} and B (2)

B is always taken (unconditional).
B{cc} is only taken if the condition is true.

b) Explain what happens to the PC when an unconditional branch instruction is executed compared to when an instruction other than a branch is executed. (2)

When a non-branch instruction is executed the PC is incremented (usually by 2) to point the next instruction in memory.
An unconditional branch will increment or decrement the PC by an arbitrary amount determined by the branch.

c) What does the compiler do when compiling an instruction like: LDR R5, =0xAA (2)

- places the word 0xAA later in flash
- generates a PC relative load to that word

d) Which is preferable? Why? MOVS R5, #0xAA vs LDR R5, =0xAA (2)

MOVS is preferable. The 0xAA is immediate data so does not take up extra room in flash.

Also but less prefered answer: MOVS takes one cycle while LDR takes two.

Question 6:

a) Give an example of two dev board peripherals and two internal peripherals. (4x 0.5)

Internal: ADC, GPIO

Dev board: LCD, LEDs

b) Consider the attached appendices. Write a block of code which will set bits 0 and 1 of the GPIOC_OTYPER while leaving all other bits unchanged. (3)

```
LDR R0, =0x48000800
LDR R1, [R0, #0x04]
MOVS R2, #0x03
ORRS R1, R1, R2
STR R1, [R0, #0x04]
```

c) What is the purpose of the GPIOx_PUPRD? Why is the functionality it provides necessary? (2)

Enables or disables pull up and pull down resistors.
Necessary in order to define default level when pin left floating.

Question 7

a) Assume the following block of code has run. (5)

```
MOV R0, #0xAA
MOV R1, #0xBB
CMP R0, R1
```

Remember: A CMP implements: Rn + inverse(Rm) + 1
Will the following flags be set, cleared or unchanged? Explain why. Marks for explanation. (2 mark for N flag. 3 marks for V flag.)

N:
0xAA + Inverse(0xBB) + 1 = 0xFFFFFFEF.
This has the msb set. msb is N, hence N = 1

V:
As shown above, calculation is 0xAA + 0xFFFFFF44.
When treated as signed numbers, this is equal to a small positive plus a small negative which does not cause overlfow.
No signed overflow, hence V = 0

b) Will the following branch be taken? Why or why not? (mark for reason) (2)

```
MOVS R0, #0xF7
MOVS R1, #0x08
ANDS R0, R0, R1
BEQ foo
```

Result of ANDS is 0.
Hence Z = 1
BEQ taken when Z=1, hence branch taken.

c) How many CPU cycles does the following block take to execute? (3)

```
        MOVS R0, #100          (1 cycle)
loop:   SUBS R0, R0, #2        (1 cycle)
        CMP R0, #0             (1 cycle)
        BNE loop               (3 cycles if taken, 1 cycle of not taken)
```

50 loop iterations.
1 + 50(1 + 1 + 3) = 251
On last iteration, BNE not taken, hence two cycles less:
249

d) Assuming a 48 MHz clock, how much time does the above code take to run? (1)

0,000 005 188 seconds, 5.1875 us

e) When an exception occurs (other than a reset exception) what 3 steps does the CPU take to handle the exception? (3)

- stack current state
- fetch vector from table
- load PC with vector and continue execution.

Question 8

a) What two things happen when a POP operation is performed? Specify them in the order which they occur. (2)

- load register whith whatever SP is pointing to
- increment SP by 4

b) Assume the following block of code has just executed:
PUSH {R0}
PUSH {R3}
PUSH {R7}
PUSH {R4}
Write a single instruction which will load R0 with whatever value was in R7 when R7 was pushed to the stack. **The instruction should not modify the SP.** (2)

```
LDR R0, [SP, #4]
```

c) Explain the difference between B and BL (2)

BL stores the address of the next instruction to be executed in the link register.

d) Discuss reasons why subroutines are useful. (2)

maximises code reuse which:

- minimises memory footprint
- allows for easier debugging as code only has to be modified in one place.

Question 10

a) What is an Analogue to Digital Convertor? (2)

A peripherals which makes high resolution approximations of voltages.

b) Assume a 8 bit ADC running off of 3.3 V.
What voltage does a reading of 0xF0 correspond to? (2)

(0xF0 / 255) * 3.3 = 3.09 V

Dividing by 256 was also marked as correct.

c) Consider the appendices. Assuming the ADC has:
ADC_CFGR1_ALIGN = 1 and
ADC_CFGR1_RES = 3
What value will be present in the ADC_DR assuming it is digitising an applied voltage of 1 V and running off of a 3 V supply rail. (3)

RES = 6 bits. = max of 63
(⅓) * 63 = 21

Data starts at bit 2. Bit shift of 2 = x4
21 x 4 = 84 = 0x54

Bonus

Describe the electrical specifications of the bus used to interface with the external temperature sensor. (3)