Digital Systems: Sample exam questions

Mike Spivey, Trinity Term, 2019

1 [2011/2] In answering this question, it does not matter if you use instructions which are not entirely identical to those of Thumb code, provided their meaning is clear and they are essentially similar to those which can be provided by the standard processor used in the course.

(a) For each of the following conditions, write a short seqence of instructions that branch to label `skip` if the condition is true: (i) the value of register `r0` is zero, (ii) the value is non-zero, (iii) the value is odd, (iv) the value is even. In each case, you may assume that register `r3` may be overwritten. [4 marks]

(b) Suppose that `count` is a routine which returns the number of set bits (bits equal to 1) in its argument. Write pseudocode for a routine `acount` which takes an array `a` and a number `n` and returns the number of set bits in the first `n` elements of `a`. [2 marks]

(c) Suppose that `count` is the label of a Thumb subroutine which returns the number of set bits in its argument.

Use this as a subroutine to implement in Thumb code another subroutine `acount` which, given the address and length of an array of words, counts the total number of set bits in that array.

Use (and assume that `count` uses) the standard Thumb calling convention: that registers `r4`–`r7` are preserved across calls, the arguments are passed in `r0`–`r3`, the result returned in `r0`, and the `r0`–`r3` are scratch registers. The stack pointer `sp` decreases as the stack grows, and the return address register is `lr`. [6 marks]

(d) Calculate the value of `0x358 & 0x357` where the operator `&` means the bitwise *and* of corresponding bits of the arguments, and explain how the number of set bits in the value of $x \,\&\, (x-1)$ is related to the number of set bits in x. [2 marks]

(e) Outline in pseudocode a program to count the number of set bits in a variable x, taking time proportional to the answer. Translate your code into Thumb instructions to implement the routine `count` mentioned in part (c). [4 marks]

(f) Had you known exactly what code you would write for `count` in part (e), in what ways could you have made the code in part (c) simpler?
[2 marks]

2 [part of 2008/3] This question concerns assembly-language programming in Thumb code. *You need not be concerned with the details of instruction mnemonics, provided the meaning of each instruction is clear and they fit into the overall architecture of the machine.*

The following program fragment determines the maximum value in an array $a[0..N)$ of positive integers; it returns 0 if $N = 0$.

```
int i, m;
static int a[N];

i = 0; m = 0;
while (i < N) {
    if (a[i] > m) m = a[i];
    i = i+1;
}
```

A partial translation of this program fragment into Thumb code is as follows:

```
    mov r0, #0          @ i = 0
    mov r1, #0          @ m = 0
    b test
loop:
    ...                 @ lines not shown
test:
    cmp r0, #N
    blt loop
```

(a) Complete the translation of the program fragment into assembly code, keeping the variables i and m in registers. Assume that the base address A of array a occupies 32 bits, while the constant N fits into 8 bits.
[8 marks]

(b) Show how the program can be speeded up by keeping the address of $a[i]$ in a register. [4 marks]

3 [2010/2]

(a) Briefly describe the function of n-type and p-type MOS transistors, giving just enough detail of their behaviour to explain their use in the design of CMOS gates. [2 marks]

(b) Give the design of a 2-input NAND gate in CMOS, and explain why the output has the correct value for each combination of the inputs.
[5 marks]

(c) A certain application needs a combinational circuit with four inputs a, b, c, d and a single output z, such that

$$z = \neg((a \wedge b) \vee (c \wedge d)).$$

Design a circuit with this specification using only 2-input NAND gates. If each gate has a propagation delay of 2 ns, estimate the propagation delay of your design. [5 marks]

(d) The circuit from part (c) is too slow to be used in the proposed application. Design a special-purpose CMOS gate with the same specification, and estimate its propagation delay if it is fabricated using the same technology as the NAND gates in part (c). [8 marks]

4 [2018/1, modified] Design a Bit Sequence Recogniser (BSR) circuit that recognises the sequence 1101. The BSR circuit has one data input, one signal output, and one clock input line. When the given sequence has arrived on the serial data input line, the output becomes one for the next clock cycle, otherwise it is zero. The recognised sequences may overlap. You may assume that the serial input signal is synchronised with the same clock signal as the circuit. The example shows the recognition of two overlapping occurrences of 1101.

Data in	0	0	1	1	0	1	1	0	1	0	1	0
Signal out	0	0	0	0	0	0	1	0	0	1	0	0

Use as many state bits as convenient.

The questions that follow are really supplementary revision exercises on new aspects of the course, not trimmed to exam length and not provided with introductory 'bookwork' parts.

5 Exercise 3.3 asked you to write an interrupt-driven driver for a hardware random number generator, describing it as follows.

> "The NRF51822 has a hardware random number generator. When appropriately configured, it periodically generates an interrupt with handler `rng_handler`, and an eight-bit random number can then be retrieved from the device register `RNG_VALUE` before resetting the event flag `RNG_VALRDY` to zero."

In micro:bian, a driver process can call `connect(RNG_IRQ)` to connect to the interrupt.

(a) Design a micro:bian driver process for the random number generator, and write a subroutine

```
unsigned randbyte(void);
```

that can be called from client processes in order to fetch a random byte. As in the previous exercise, use a stack to hold a limited number of random values that have been generated but not used.

(b) Experience with the program indicates that clients are likely to call `randbyte()` several times to fetch multiple random bytes. Explain why doing so may cause an unacceptably high overhead, and show how to modify the program in order to provide a subroutine

```
unsigned randint(void);
```

that returns a random four-byte integer more efficiently than using four calls to `randbyte`.

6 The Nordic processor die has a temperature sensor that can be activated when necessary by setting `TEMP_START = 1`, and interrupts some time later

on TEMP_IRQ with TEMP_DATARDY = 1; the temperature reading (in units of 1/4 Celsius) can then be fetched from the register TEMP_TEMP.

Design a driver process for the sensor, suitable for use with up to five client processes, which perhaps may wish to search for primes less assiduously or flash the beating heart less passionately, if the processor is getting too hot for comfort. When no client process is requesting the temperature, the sensor should remain idle. As soon as a request comes, the sensor should be started, and any other requests that come in subsequently should receive the same reading when it is ready. Later requests should result in a fresh reading.

If the same reading is being sent to several client processes, and one of them should have exited by the time the reading is ready, discuss the consequences for the future functioning of the program.

7 (a) A bit-serial adder is a sequential circuit with two inputs a and b and a single output z. Successive binary digits of two numbers are presented at the two inputs at the rate of one pair of digits per clock cycle. The output of the circuit carries successive binary digits of the sum of the two numbers, with each digit output in the clock cycle immediately following the one in which corresponding input bits arrived. Design an implementation of this circuit.

(b) A bit-serial comparator takes two inputs using the same conventions as the adder, but produces three outputs u, v and w, one active at a time, with $u = 1$ if the number so far presented at a is greater than that presented at b, and $v = 1$ if the two numbers are equal, and $w = 1$ if the number as a is less than that at b. Design an implementation of this circuit too.