HERIOT WATT UNIVERSITY

SCHOOL OF MATHEMATICAL AND COMPUTER SCIENCES

Computer Science

F28HS

HARDWARE-SOFTWARE INTERFACE

Semester 2 2015/16

Duration: Two Hours

ANSWER THREE QUESTIONS

Answer each question in a separate script book.

1. Consider the following C program fragment:

```
 1. struct link {  int val; struct link * left, * right;  };

 2. struct link * newL(int v)
 3. {  struct link * l;
 4.    l = (struct link *)malloc(sizeof(struct link));
 5.    l->val = v;
 6.    l->left = NULL;
 7.    l->right = NULL;
 8.    return l; }

 9. addL(struct link ** l,int v)
10. {  struct link * li;
11.    li= newL(v);
12.    if(*l==NULL)
13.     *l = li;
14.    else
15.    {  (*l)->left = li;
16.       li->right = *l;
17.       *l = li; }
18. }
...
19. struct link * l;
20. l = NULL;
21. addL(&l,1);
22. addL(&l,2);
23. addL(&l,3);
24. printf("%d %d %d\n",i->val,i->right->val,i->right->right->val);
```

(a) How many bytes are required to store:
(i) `struct link *`
(ii) `struct link`
(iii) `struct link **` (4)

(b) Explain what happens on lines 19-24, refering to lines 2-8 and 9-18 as appropriate. Illustrate your answer with appropriate diagrams to show how `l` changes. (12)

(c) Briefly outline the differences in memory management between C and Java. (4)

2. Consider the following ARM assembly language program fragment:

```
 1. .global _start

 2. _start:
 3.     LDR R1, =A
 4.     MOV R2, #MAX
 5.     MOV R3, #0
 6. loop:
 7.     CMP R2, #0
 8.     BEQ end
 9.     LDR R4, [R1]
10.     ADD R3, R4
11.     ADD R1, #4
12.     SUB R2, #1
13.     B loop
14. end: ...

15. .data
16. .equ MAX, 5
17. A:  .rept MAX
18.     .word 0x00
19.     .endr
```

(a) Why is `.global start` necessary? (2)

(b) Briefly explain the purposes of the instructions on lines 3, 4, 7, 8, 9, 10, 11, 12 and 13. (10)

(c) Why is 4 added to R1 (line 11) but 1 is subtracted from R2 (line 12)? (2)

(d) What does the program fragment do? (2)

(e) Briefly discuss the view that there is no longer any need to understand assembly language programs. (4)

3. In the course we discussed the importance of an architecture's *memory hierarchy* for the performance of a program.

(a) Explain in words the notion of a "cache" in modern architectures and the role it plays in making memory access more efficient. (3)

(b) Explain in words the notions of "cache-hit" and "cache-miss". Discuss the impact that a cache-miss has on the performance of a program. (3)

(c) In the program below, assume that variables `x` and `y` map to the same location in a direct-mapped cache. Will the cache performance of the program be good or bad? Suggest at least one way (either in software or hardware) how to improve the cache performance.

```
for (i=0; i<1000000; i++)
  sum += i*(x+y)
```

(6)

(d) In a computational science application, the distance between points in a 3D space needs to be computed. For two given points with coordinates (x, y, z) and (x', y', z'), their distance is defined as follows: $\sqrt{(x-x')^2 + (y-y')^2 + (z-z)^2}$. One compute-intensive function, that needs to be implemented efficiently, takes as arguments one reference point `pt` and a set of points `arr`, and should compute the sum over all distances of the points in `arr` from point `pt`. Two implementations of this function have been suggested (below), using different data structures for the set of points. Discuss in words which of these two versions is more efficient, with a particular focus on the efficiency of memory accesses and the use of the cache in the two versions.

Version 1 (using an array of points, where each point is a 3-element structure `struct_t`):

```
typedef struct {
  ulong x;  ulong y;   ulong z;
} struct_t;

// use an array-of-struct representation
static inline double distance_aos(struct_t pt, ulong n,
                                  struct_t *arr) {
  int i;
  double  sum = 0.0;

  for (i = 0; i < n; i++)
    sum += sqrt(sqr(arr[i].x-pt.x)+
                sqr(arr[i].y-pt.y)+
                sqr(arr[i].z-pt.z));
  return sum;
}
```

Version 2 (using a structure of 3 arrays, for the x, y, z coordinates, respectively):

```
// struct-of-arr data structure
typedef struct {
  ulong *xs;
  ulong *ys;
  ulong *zs;
} soa_t;

// use a struct-of-array representation
static inline double distance_soa(struct_t pt, ulong n,
                                  soa_t *s) {
  int i, j;
  double  sum = 0.0;

  for (i = 0; i < n; i++)
    sum += sqrt(sqr(s->xs[i]-pt.x)+
                sqr(s->ys[i]-pt.y)+
                sqr(s->zs[i]-pt.z));
  return sum;
}
```

Hint: Note that the two versions use different data layouts. For a set of 3 points, the layout for Version 1 is $x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3$, i.e. it starts with all coordinates of point 1 followed by those of point 2 etc; the layout for Version 2 is $x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3$, i.e. it starts with all x-coordinates of the set of points followed by all y-coordinates etc. (8)

4. In the course we discussed in detail how the GPIO pins on a Raspberry Pi 2 can be used to program external devices.

(a) In order to access an external device, such as an LED, you need to access the memory attached to the GPIO pins. However, this memory is not usually available for user-space programs under an operating system such as Raspbian.

(i) Which library function do you need to call in order to make the memory available?

(ii) Explain what the library function achieves.

(iii) Can you use this function both from C and Assembler programs?

(6)

(b) In order to control a GPIO device, such as an attached LED, on the Raspberry Pi 2, you want to conceptually "send one bit of information" to the device in order to turn it on/off. When programming a GPIO device, however, you don't use a "send" operation, but rather an existing language construct.

Answer the following questions (no code needs to be given; refer to the table at the end of the paper for the list of relevant registers):

(i) Assume your LED is connected to pin 23. What is the name of the register and the bits to write to in order to set the *mode* for this pin?

(ii) Assume your LED is connected to pin 23. What is the name of the register and the bit to write to in order to *turn on* the LED?

(iii) Assume your LED is connected to pin 23. What is the name of the register and the bit to write to in order to *turn off* the LED?

(8)

(c) The following C code snippet aims to only turn on an LED, attached to a GPIO pin with the number `PIN`, but fails in doing so. This pin has correctly been configured as an output device. The register to access has been correctly identified as `gpset`, and `gpio` is the correct base address of the GPIO registers. Identify the problem and modify the right-hand-side of the assignment to fix this problem.

```
// gpset has been correctly identified as the register to write to;
// PIN is the number of the GPIO pin
*(gpio + gpset) = PIN ;
```

(6)

END OF PAPER

REGISTER TABLE

Address	Field Name	Description	Size	Read/ Write
0x 7E20 0000	GPFSEL0	GPIO Function Select 0	32	R/W
0x 7E20 0000	GPFSEL0	GPIO Function Select 0	32	R/W
0x 7E20 0004	GPFSEL1	GPIO Function Select 1	32	R/W
0x 7E20 0008	GPFSEL2	GPIO Function Select 2	32	R/W
0x 7E20 000C	GPFSEL3	GPIO Function Select 3	32	R/W
0x 7E20 0010	GPFSEL4	GPIO Function Select 4	32	R/W
0x 7E20 0014	GPFSEL5	GPIO Function Select 5	32	R/W
0x 7E20 0018	-	Reserved	-	-
0x 7E20 001C	GPSET0	GPIO Pin Output Set 0	32	W
0x 7E20 0020	GPSET1	GPIO Pin Output Set 1	32	W
0x 7E20 0024	-	Reserved	-	-
0x 7E20 0028	GPCLR0	GPIO Pin Output Clear 0	32	W
0x 7E20 002C	GPCLR1	GPIO Pin Output Clear 1	32	W
0x 7E20 0030	-	Reserved	-	-