THE AUSTRALIAN NATIONAL UNIVERSITY

Mid Semester Examination, April 2007

COMP2300
(Introduction to Computer Systems)

Writing Period: 1 hour duration

Study Period: 0 minutes duration

Permitted Materials: One A4 page with notes on both sides.
NO calculator permitted.

Questions are NOT equally weighted.
This exam will contribute 20% to your final assessment.

The questions are followed by labelled, framed blank panels into which your answers are to be written. Additional answer panels are provided (at the end of the paper) should you wish to use more space for an answer than is provided in the associated labelled panels. If you use an additional panel, be sure to indicate clearly the question and part to which it refers to.

The marking scheme will put a high value on clarity so, as a general guide, it is better to give fewer answers in a clear manner than to outline a greater number in a sketchy, half-answered fashion. The Appendix contains information on the PeANUt instruction set, as well as a table with powers of 2 values in decimal.

Please write clearly – if we cannot read your writing you may lose marks!

Name (family name first):

Student Number:

Official use only:

Q1 (13)	Q2 (17)	Total (30)

QUESTION 1 [13 marks]

(a) Assume memory addresses `0x02000411` to `0x02000415` contain the following 8-bit binary values:

Address	`0x02000411`	`0x02000412`	`0x02000413`	`0x02000414`	`0x02000415`
Binary value	`00100010`	`10001101`	`11001000`	`01100011`	`00110000`

Assume `sizeof(x) = 2` and `sizeof(y) = 2`, `&x = 0x02000412` and `&y = 0x02000414`, and the data storage is *big endian*.

(i) What would be printed by the following C statement?

```
printf("Values for x+y:  %x %o", x+y, x+y);
```

Clearly show how you derive your answers.

QUESTION 1(a)[i] **[2 marks]**

(ii) What would be printed by the following C statement?

```
printf("Values for x+y:  %d %u", x+y, x+y);
```

Clearly show how you derive your answers.

QUESTION 1(a)[ii] **[2 marks]**

Question 1 (continued)

(b) The IEEE single-precision floating-point standard is: 1 bit sign, 8 bits exponent with a bias of 127, and the remaining 23 bits are the mantissa (with an implicit leading bit). To **four decimal digits**, what floating point number is represented by the following 32-bit number (given in hexadecimal representation):

`0xC1B80000`

QUESTION 1(b) **[2 marks]**

(c) Consider the following C code:

```
char c1=0; unsigned char c2=0;
while (1) {
  c1++; c2++;
}
```

Describe the patterns of values of `c1` and `c2` 9in decimal) that would be generated by the iterations of the loop, as it endlessly executes.

QUESTION 1(c) **[1 mark]**

Question 1 (continued)

(d) Name two limitations of floating point arithmetic that might adversely affect the accuracy of any numerical calculations carried out with it.

What is the underlying reason that is common to both these limitations and the phenomenon associated with the integer arithmetic illustrated in Question 1(c)?

QUESTION 1(d) **[2 marks]**

(e) Briefly describe the functions of the main components of the Central Processing Unit (CPU) of a typical computer.

QUESTION 1(e) **[2 marks]**

(f) State Moore's Law. Briefly describe its impact on computer technology over the last 50 years.

QUESTION 1(f) **[2 marks]**

Student Number: .....

QUESTION 2 [17 marks]

(a) The following program manipulates character strings from the standard C library.

```
#include <stdio.h>
#include <string.h>
int main(){
  char str1[100] = "computer";
  char str2[100] = " systems";
  char *str3;

  str3 = strcat(str1, str2);
  printf("str1 (length=%2d) is: %s\n", strlen(str1), str1);
  printf("str2 (length=%2d) is: %s\n", strlen(str2), str2);
  printf("str3 (length=%2d) is: %s\n", strlen(str3), str3);

  return 0;
}
```

Relevant details of `strlen()` and `strcat()` functions are:

```
int strlen(const char *s);
// returns the length of the string s,
// not including the terminating '\0' character


char *strcat(char *dest, const char *src);
// appends the src string to the dest string, overwriting the
// '\0' character at the end of dest, and then adds  a
// terminating  '\0'  character. It returns the value of dest
```

(i) Write the output that the program would generate when it is compiled and run.

QUESTION 2(a)[i] **[2 marks]**

Question 2 (continued)

(ii) Write an implementation of the function `strcat()` as defined above. Your code must not call any other function except `strlen()`.

QUESTION 2(a)[ii] **[6 marks]**

(b) A C program `myprogram.c` has its main program function declared as `int main (int argc, char *argv[]);`.

(i) Describe in words the meaning of the declaration `char *argv[]`.

QUESTION 2(b)[i] **[1 mark]**

(ii) State the significance of the declaration, in the context when `myprogram.c` is compiled into an executable program, i.e. with `gcc -Wall -o myprogram myprogram.c`.

QUESTION 2(b)[ii] **[2 marks]**

Question 2 (continued)

(c) Briefly state the purpose of a function prototype ('header') and explain how these can support separate compilation of libraries.

QUESTION 2(c) **[2 marks]**

(d) In C, function parameters are *passed by value*. Explain, in terms of a simple example, how an integer variable may be effectively *passed by reference* in C.

QUESTION 2(d) **[1 mark]**

(e) Briefly describe the difference between the following two ways of declaring a constant to represent π: `#define PI 3.142` and `const double PI = 3.142;`.

QUESTION 2(e) **[1 mark]**

Question 2 (continued)

(f) In C, storage for an integer array 'a' of length 100, may be allocated *statically*, e.g. by `int a[100];`', or *dynamically*,
e.g. by `int *a; a = (int *) malloc(100 * sizeof(int));`.
Name one pitfall (common programming error) that is common to both static and dynamic allocation, and one pitfall that is unique to dynamic allocation. State a circumstance where dynamic allocation is preferable over static.

QUESTION 2(f) **[2 marks]**

Additional answers to QUESTION __(__)[__]

Additional answers to QUESTION __(__)[__]

Student Number: .....

Additional answers to QUESTION __(__)[__]

Additional answers to QUESTION __(__)[__]

Appendix

x	2^x
-5	0.03125
-4	0.0625
-3	0.125
-2	0.25
-1	0.5
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096
13	8192
14	16384
15	32768
16	65536

Table 1: Powers of 2 in decimal