THE UNIVERSITY OF WAIKATO
Department of Computer Science

COMP201Y — Computer Systems
Exercise 3 Test — 12th May 2003

Worth 10% — Marked out of: 30

Time allowed: 45 Min

---

1. Figure 1 shows the format for an Internet Protocol (IP) Datagram header. Figure 2 shows a hex dump of the start of an IP datagram. Using the header format, determine the IP Version and Time To Live in decimal, and the Source and Destination IP Addresses in hexadecimal, of the datagram. You should show any working. Notes: For each 32 bit word in Figure 1, the most-significant bit is on the left. The hex dump uses a Big Endian byte ordering.

(4 marks)

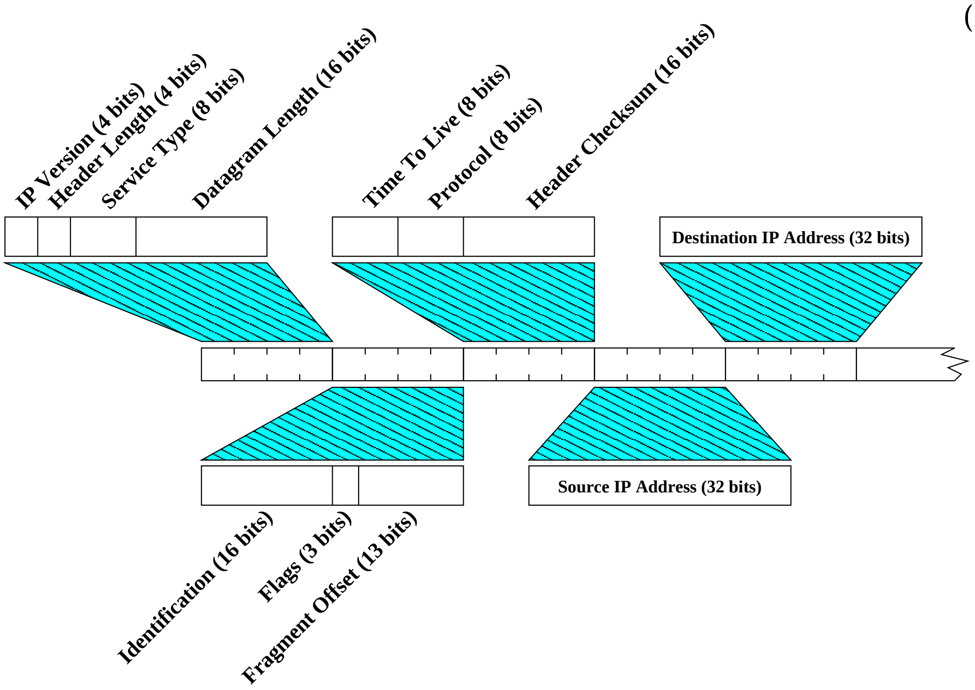

Figure 1: IP Datagram header format

```
00000000: 4500 05dc d450 4000 4006 6689 82d9 fa0d  E....P@.@.f.....
00000010: 82d9 fa81 b43d 1771 9947 cba6 9a1f dab7  .....=.q.G......
00000020: 8010 7c70 af42 0000 0101 080a 07b4 d79b  ..|p.B..........
00000030: 006a 61b7 2700 3230 3230 2020 2020 2020  .ja.‘.2020
00000040: 2020 2020 2020 2020 2020 2020 2020 2020
```

Figure 2: HEX dump of IP datagram header

2. Convert the number -17.8125 to 32bit IEEE-754 floating point format. The format for a 32bit IEEE-754 floating point number is given in Figure 3. Show all working.

(3 marks)

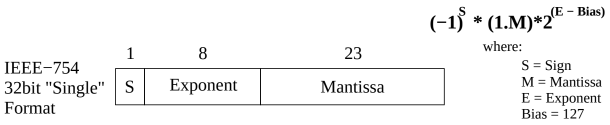

Figure 3: 32bit IEEE-754 floating point format

3. Figure 4 contains a small WRAMP assembly language program which a programmer has been asked to check. The program has assembled, linked and uploaded to a REX board. The `WRAMPmon` breakpoint facility has been used to set a breakpoint in the middle of a loop (line 11 in Figure 4) so that the programmer can check the program is executing correctly. Before running the program the programmer decides to disassemble the program using the "`dis`" command. The disassembled program including the breakpoint location is shown in Figure 5.

   The programmer then uses the `go` command to start the program executing. Each time the breakpoint is encountered the contents of the relevant registers are noted then the programs execution is continued using the `cont` command. This continues until the programs execution completes.

   (a) On the answer sheet provided, note what the contents of registers three ($3), four ($4) and five ($5) would have been each time the breakpoint was encountered. You will need to trace the execution of the code and provide the contents for all three registers once for each time the breakpoint would have been encountered.

   (6 marks)

   (b) This program performs a computation on the value in register $3 to produce a final result in register $4. What characteristic of the value that is in register $3, does the result represent?

   (2 marks)

```
1:    .global main
2:    main:
3:        addi    $3, $0, 0x095d
4:        addi    $4, $0, 0
5:    loop:
6:        beqz    $3, finished
7:
8:        addi    $4, $4, 1
9
10:       subi    $5, $3, 1
11:       and     $3, $3, $5
12:
13:       j       loop
14:   finished:
15:       j       exit
```

Figure 4: WRAMP code to check

```
0x00000 1300095d       addi     $3,$0,0x095d
0x00001 14000000       addi     $4,$0,0x0000
0x00002 a0300004       beqz     $3,0x00007
0x00003 14400001       addi     $4,$4,0x0001
0x00004 15320001       subi     $5,$3,0x0001
0x00005 033b0005 !BRK! and      $3,$3,$5
0x00006 40000002       j        0x00002
0x00007 4000002b       j        0x0002b
```

Figure 5: WRAMPmon disassembly of the uploaded code

4. You are required to write a function in WRAMP assembler. This function is to be of the form specified by the following function prototype.

   `int power(int base, int exponent);`

   This function should compute the result of raising `base` to the power of `exponent`. It must do this in a recursive manner, using the algorithm shown in the psuedo-code in Figure 6. You must ensure that the function you write complies to the WRAMP register use and subroutine conventions.

   (8 marks)

```
power(base, exponent) {
   if (exponent = 0) then
         return 1
   else
         return base * power(base, exponent - 1)
}
```

Figure 6: Psuedo-code for the recursive power function.

5. Figure 7 shows a function written in C that is in a file called `myfunc.c`. This file is compiled with the WRAMP C Compiler using the command '`wcc -S myfunc.c`', to generate a file called `myfunc.s`. Figure 8 shows an incomplete listing of `myfunc.s`, where code corresponding to certain lines within `myfunc.c` have been removed at the points indicated.

   (a) List the line numbers of the lines within the C code for which the corresponding assembler code has been removed.

   (2 marks)

   (b) Insert instructions into the assembler listing to make it a complete and correct translation of the C code.

   (5 marks)

```
 1:   int lookup(int index);
 2:   void printnum(int num);
 3:
 4:   void myfunc(int start, int count)
 5:   {
 6:     int i, end, sum;
 7:
 8:     sum = 0;
 9:     end = start + count;
10:
11:     for (i = start ; i <= end ; i++)
12:       sum = sum + lookup(i);
13:
14:     if (sum < 0)
15:       sum = -sum;
16:
17:     printnum(sum);
18:   }
```

Figure 7: myfunc.c

```
     .global myfunc
     .text
     myfunc:
         subui   $sp, $sp, 7
         sw      $5, 1($sp)
         sw      $6, 2($sp)
         sw      $7, 3($sp)
         sw      $12, 4($sp)
         sw      $13, 5($sp)
         sw      $ra, 6($sp)
         addu    $6, $0, $0
         lw      $13, 7($sp)
         lw      $12, 8($sp)

# Instructions have been removed here.

         addu    $7, $0, $13
         j       L.5
     L.2:

# Instructions have been removed here.

     L.3:
         addi    $7, $7, 1
     L.5:
         sle     $13, $7, $5
         bnez    $13, L.2
         sge     $13, $6, $0
         bnez    $13, L.6
         subu    $6, $0, $6
     L.6:

# Instructions have been removed here.

     L.1:
         lw      $5, 1($sp)
         lw      $6, 2($sp)
         lw      $7, 3($sp)
         lw      $12, 4($sp)
         lw      $13, 5($sp)
         lw      $ra, 6($sp)
         addui   $sp, $sp, 7
         jr      $ra
```

Figure 8: myfunc.s

Department of Computer Science
University of Waikato

COMP201Y — Computer Systems

Exercise 3 Test 2003
Answer Sheet

If you need more space than is provided, write on the reverse of the page and clearly indicate this.

Name: ID Number:

1. Using the diagram provided in Figure 1 determine the IP Version and Time To Live in decimal, and the Source and Destination IP Addresses in hexadecimal, from the datagram header hexdump in Figure 9. You should show any working.

(4 marks)

```
00000000: 4500 05dc d450 4000 4006 6689 82d9 fa0d  E....P@.@.f.....
00000010: 82d9 fa81 b43d 1771 9947 cba6 9a1f dab7  .....=.q.G......
00000020: 8010 7c70 af42 0000 0101 080a 07b4 d79b  ..|p.B..........
00000030: 006a 61b7 2700 3230 3230 2020 2020 2020  .ja.‘.2020
00000040: 2020 2020 2020 2020 2020 2020 2020 2020
```

Figure 9: HEX dump of IP datagram header

IP Version	□	Source IP Address	□
Time To Live	□	Destination IP Address	□

2. Convert -17.8125 to a 32bit IEEE-754 floating point value. Show all working.

(3 marks)

3. (a) What are the contents of the following registers each time the breakpoint is encountered? You should fill in one line each time you believe a breakpoint was hit. You should not need more lines than are provided, although you may need less.

(6 marks)

$3	**$4**	**$5**

(b) What function does this program perform?

(2 marks)

4. Write the power function to recursively compute powers in the following space.

(8 marks)

```
.global power
.text
power:
```

```
        jr      $ra
```

5. (a) List the line numbers of the lines within the C code for which the corresponding assembler code has been removed.

(2 marks)

(b) Insert instructions into the assembler listing to make it a complete and correct translation of the C code.

(5 marks)

```
.global myfunc
.text
myfunc:
    subui   $sp, $sp, 7
    sw      $5, 1($sp)
    sw      $6, 2($sp)
    sw      $7, 3($sp)
    sw      $12, 4($sp)
    sw      $13, 5($sp)
    sw      $ra, 6($sp)
    addu    $6, $0, $0
    lw      $13, 7($sp)
    lw      $12, 8($sp)



    addu    $7, $0, $13
    j       L.5
L.2:






L.3:
    addi    $7, $7, 1
L.5:
    sle     $13, $7, $5
    bnez    $13, L.2
    sge     $13, $6, $0
    bnez    $13, L.6
    subu    $6, $0, $6
L.6:





L.1:
    lw      $5, 1($sp)
    lw      $6, 2($sp)
    lw      $7, 3($sp)
    lw      $12, 4($sp)
    lw      $13, 5($sp)
    lw      $ra, 6($sp)
    addui   $sp, $sp, 7
    jr      $ra
```