

Blankpageforworking

## CSE 331 Part B – Introduction to Computer Systems

### Question B1 [1+2=3 marks]

This question is about bitwise operations.

Consider the following code:

```
/* Mode bits for opening files. */
#define MODE_READ      1
#define MODE_WRITE     (1 << 1)
#define MODE_APPEND   (1 << 2) /* Open at end of file. */
#define MODE_BINARY   (1 << 8)
```

```
int myOpen(const char *filename, const int mode);
```

The `myOpen` function's second parameter contains a value formed by a bitwise OR (`|`) of the defined values. For instance:

```
x = myOpen("datafile", MODE_READ | MODE_WRITE);
```

means to open `datafile` for both reading and writing.

a. Write the following values in decimal.

MODE\_READ = \_\_\_\_\_

MODE\_WRITE = \_\_\_\_\_

MODE\_APPEND = \_\_\_\_\_

MODE\_BINARY = \_\_\_\_\_

MODE\_WRITE | MODE\_APPEND = \_\_\_\_\_

MODE\_READ | MODE\_BINARY = \_\_\_\_\_

b. Using the `mode` parameter, how could the `myOpen` function determine if a file should be opened

(i) for reading?

if ( \_\_\_\_\_ )

(ii) for reading but not writing?

if ( \_\_\_\_\_ )

Write an expression between the brackets in each case. Note that your above answer must work independently of any `MODE_BINARY` or `MODE_APPEND` bits.

3

Blankpageforworking

**Question B2 [1+1+1=3 marks]**

This question is about the MIPS instruction set.

a. Explain why five bits are sufficient to identify a MIPS register in an instruction.

b. Explain the difference between the `li` (load immediate) and `move` instructions. Give an example of each to illustrate the difference.

c. `li` (load immediate) and `move` are pseudoinstructions. Rewrite your two example instructions from (b) above, using only real machine instructions.

3

**Question B3 [0.5+0.5+1+2=4 marks]**

The following question refers to the C code on the opposite page.

a. What does this program do? (Don't just describe the algorithm.)

b. If given the input 35, what does the program print?

c. Draw a diagram of the stack as it would appear after the line marked **★**.  
Include in your diagram:

- the values of `$fp` and `$sp`, and
- the values, addresses and names of all relevant words on the stack.
- arrows showing to which words `$sp` and `$fp` point.

Assume that the user input is 35, and that the value of `$sp` prior to the program's beginning is `0x7FFF0000`.

d. Complete the partial MIPS program beside the C code so that it remains a faithful translation of the original C program.

2

<code>#include &lt;stdio.h&gt;</code>	
<code>int main()</code>	<code>.text main:</code>
<code>{   int num;   int fact;</code>	<code>  move \$fp, \$sp   subu \$sp, \$sp, 8</code>
<code>  scanf("%d", &amp;num);</code>	<code>  li \$v0, 5   syscall   sw \$v0, -8(\$fp)</code>
<code>  fact = num - 1;</code>	<code>  lw \$t0, -8(\$fp)   sub \$t0, \$t0, 1   sw \$t0, -4(\$fp)     ★</code>
<code>  while (fact &gt; 1)   {</code>	<code>loop:</code>
<code>    if (num % fact == 0)</code>	
<code>    {       printf("%d", fact);</code>	
<code>    break;     }</code>	<code>  j endlp</code>
<code>    fact = fact - 1;</code>	<code>  skip: lw \$t0, -4(\$fp)       sub \$t0, \$t0, 1       sw \$t0, -4(\$fp)</code>
<code>  }</code>	<code>  j loop</code>
<code>}</code>	<code>endlp: li \$v0, 10       syscall</code>