School of Computer Science

Systems and Architecture (AE1SYS) Autumn 2016

Assembly Programming Coursework

Deadline: Monday 21st of November, 2016

Collaborating in small groups of up to three students is permitted, but you must implement your own programs (absolutely *do not* copy and paste from others) and provide your own answers where appropriate.

Note that lacking proper comments and user prompts will lose mark.

Submit your UNCOMPRESSED assembly program files to Moodle.

1. Write a program in MIPS32 assembly language which reads a number n from the console, and prints out the factorial of n:

   $$n! = n \cdot (n-1) \cdot (n-2) \cdots 2 \cdot 1.$$

   The procedure in Java, given n, might look like:

   ```
   int f = 1;
   for(int i = 1; i <= n; i++){
        f = f * i;
   }
   ```

   (15 marks)

2. Implement a program which prompts user two integers inputs x, y from the console and calculate the following expression in signed 32-bit arithmetic:

   $$x^2 + 9y^2 + 6xy - 6x - 18y + 9$$

   Note that you are NOT allowed to use pseudo-instructions with overflow checking for the calculation (i.e. you can not use `mulo`). If an overflow occurs during any step of the calculation, you should print an error message instead, and stop the program.

   Hint: You could simplify the expression before calculation. Please remember to test your program with a range of different inputs, e.g. $x = 2, y = 3$; $x = -3, y = 4$; $x = 1\,000, y = 150\,000 \ldots$

   (25 marks)

3. Implement the following specification of the `strchr` function which, given an ASCII character code and the starting address of a string, returns the offset of the first occurrence of the character in the string, or `-1` if the character cannot be found:

```
int strchr(char needle, char[] haystack){
    for(int i = 0; haystack[i] != NUL; i++)
        if(haystack[i] == needle)
            return i;
    return -1;
}

void main(void){
    int offset;
    char haystack[256];
    char needle[2];
    haystack <- read_string;  // $a0 == haystack; $a1 == 256
    needle <- read_string;    // $a0 == needle; $a1 == 2
    offset = strchr(needle[0], haystack);
    if(offset >= 0)
        printf("found at offset: %d", offset);
    else
        printf("not found");
}
```

To declare a `char buffer[`n`]` *of* n *bytes, use the* `.space` n *directive in the* `.data` *segment.*

Please implement the `main` function above as well. Insert a comment in your program to illustrate how you would use `strchr` to calculate the length of a string.

(25 marks)

4. In this exercise you will use the Newton-Raphson method to calculate the *positive* square root of a number $n \geq 0$. In summary, the method is:

$$x_0 \approx \sqrt{n}$$
$$x_{i+1} = \frac{1}{2}\left(x_i + \frac{n}{x_i}\right)$$

with $\sqrt{n} = \lim_{i \to \infty} x_i$.

Since we are working with a finite representation of floating point numbers, we can stop when the difference between x_i and x_{i+1} is small enough. The following pseudo-C program illustrates the Newton-Raphson method:

```
// input a floating point number n
float x0 = n, x1 = 0.5 * n;
while(abs(x0 - x1) > 1e-6):
        x0 = x1;
        x1 = 0.5 * (x0 + n / x0);
// The square-root of n is approximately x1
```

Implement a MIPS assembly program which implements the above method for single-precision (32-bit) floating point numbers, without using the `sqrt.s` instruction.

(35 marks)