COMP9032 Lab 2

August, 2018

1. Objectives

In this lab, you wi ming on

- memory a
- stack and

2. Tasks

2.1 Task 1: String to Integer Conversion (Due Week 5)

The C program in Figure 1 **implements the function** of converting a string to an integer. The string is given in main() and its integer is obtained by calling function atoi(). Manually translate the program into an assembly program **with *atoi* implemented as a function.** Assume the string is stored in the program memory and that an integer takes two bytes.


```
int main(void) {
        char s[ ] = "12345";
        int number;
        number = atoi(s);
        return 0;
}

int atoi(char *a) {
        char i;
        char c;
        int n;

        n = 0;
        c = *a;
        for (i=1; ((c >='0') && (c<='9') && (n<65536)); i++){
                n = 10 * n + (c -'0');
                c = *(a+i);
        }
        return n;
}
```

Figure 1 string_to_number.c

2.2 Task 2: Positional Division (Due Week 6)

Hand-division is a positional division that uses a series of left shifts, magnitude checks and multiple subtractions to get the result. The program in Figure 2 describes the positional division algorithm for the 16-bit **binary division**. Manually translate the program into an assembly function. Assume the dividend and the divisor are stored in

the program memory and that the quotient is saved in the data memory. To test your design, you need to create a caller to the function.


```
int posdiv(unsigned int dividend, unsigned int divisor) {
    unsigned int bit_position = 1;
    unsigned int quotient = 0;
    while ((divisor < dividend) && !(divisor & 0x8000)) {
        divisor = divisor << 1;
        bit_position = bit_position << 1;
    }
    while (bit_position) {
        if (dividend >= divisor) {
            dividend = dividend - divisor;
            quotient = quotient + bit_position;
        }
        divisor = divisor >> 1;
        bit_position = bit_position >> 1;
    }
    return quotient;
}
```


Figure 2: binary_positional_division.c

Note: Each task is worth 6 marks. All your programs should be well commented and easy to read. Up to 1 mark will be deducted for each program without proper and sufficient comments.