NUS ECE CG2028 Computer Organization

Assignment 1 – Semester 1, AY2024/25

ARM v7-M Assembly Language and C Programming

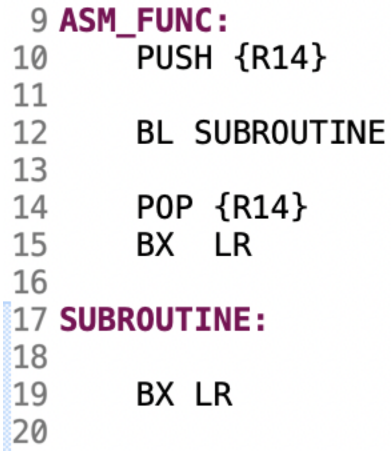
The questions in this assignment manual are based on the Assign1 template.

- 50% towards final grade, 50 marks in total
    - Code: 20 marks
        - 4 given test cases - 2 marks per each case
        - 2 hidden test cases - 2 marks per each case
        - 3 marks for coding optimisation and style
        - 5 marks for machine codes
    - Report: 30 marks
        - 16 marks for Q&A
        - 5 marks for microarchitecture design
        - 4 marks for program logic
        - 5 marks for discussions of the improvements
    - Peer evaluation: only if necessary

Q1: Knowing the starting address of `arr[]`, where `arr[]` is an array of size M. How to determine the memory address of the A-th data point, say A ≤ M? Drawing or equation can be used to explain your answer. *(1 marks)*

Q2: Compile the "EE2028_Assign1" project and execute the program. Comment the `PUSH {R14}` and `POP {R14}` lines in `asm_func()`, recompile and execute the program again.

1. Describe what you observe in (i) before and (ii) after removing the two lines and explain why there is a difference. *(1 marks)*
2. Explain the possible disadvantage(s) when using PUSH and POP instructions. *(1 marks)*

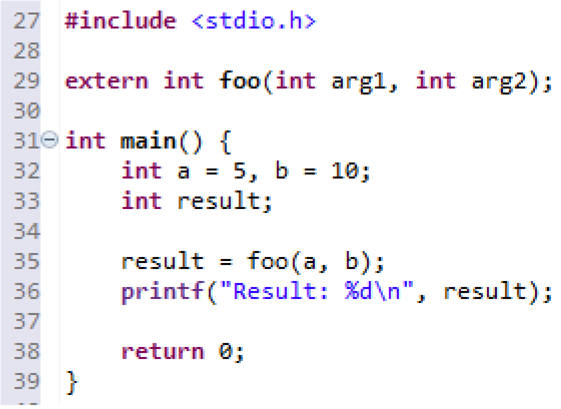


```
ASM_FUNC:
        PUSH {R14}

        BL  SUBROUTINE

        POP {R14}
        BX   LR

SUBROUTINE:

        BX LR

```

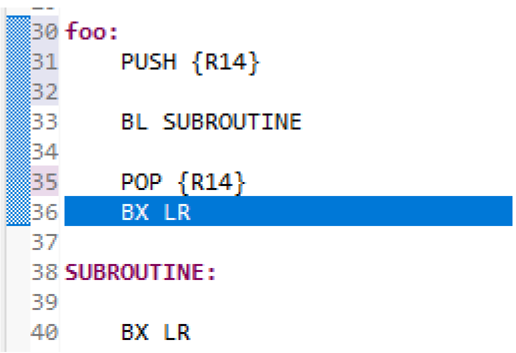
Q3: What can you do if you have used up all the general-purpose registers and you need to store some more values during processing? *(1 marks)*

Q4: Consider the following C code snippet and the corresponding assembly routine:

C Code: **Assembly Routine:**



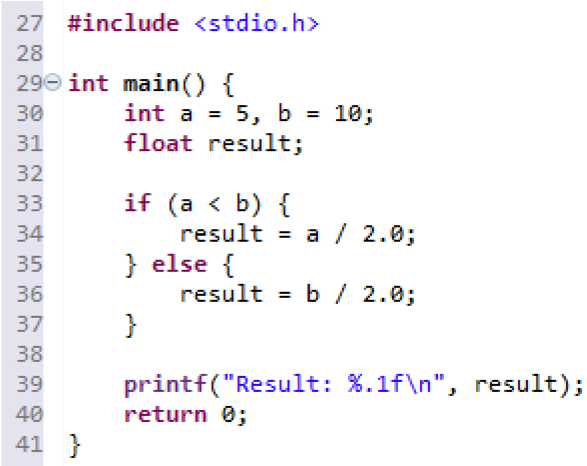
```
27  #include <stdio.h>
28
29  extern int foo(int arg1, int arg2);
30
31  int main() {
32      int a = 5, b = 10;
33      int result;
34
35      result = foo(a, b);
36      printf("Result: %d\n", result);
37
38      return 0;
39  }
```



```
30 foo:
31      PUSH {R14}
32
33      BL  SUBROUTINE
34
35      POP {R14}
36      BX LR
37
38 SUBROUTINE:
39
40      BX LR
```

Read the above code. Assume each line of the C code corresponds to a single line of assembly instruction. After executing the highlighted line (Ln 36, `BX LR`), which instruction is the Link Register (LR) pointing to at this moment? *(2 marks)*

Q5. Consider the following C code snippet:



```c
#include <stdio.h>

int main() {
    int a = 5, b = 10;
    float result;

    if (a < b) {
        result = a / 2.0;
    } else {
        result = b / 2.0;
    }

    printf("Result: %.1f\n", result);
    return 0;
}
```

1. Explain what the code is intended to do, focusing on the if-else structure. What is the output of the code? *(1 mark)*
2. Discuss how the data type of result affects the output when changing the values of `a` and `b`. What would happen if result were declared as an integer? You might need to make appropriate changes to the `printf()` function call. *(1 mark)*

You only need to submit your "optimize.s" code as a .txt file and your report as a .pdf file.

The report should not exceed 4 pages, and you don't need to paste the entire code into the report. Please include the following in your report:

- answers to the 5 questions asked in this assignment manual,
- microarchitecture design that supports MLA and MUL instructions,
- discussions of your program logic (overall logic flow, do not explain line by line),
- discussions of the improvements you have made that enhance your program efficiency (reusing registers, more efficient algorithms, etc.),
- and an Appendix that declares every member's joint and specific individual contributions towards this assignment.

==

Open Test Cases

case1

```
int a=6, b=3, c=-3;   // Polynomial coefficients
float x0 = 4.5;      // Starting point
float lambda = 0.1;   // Learning Rate
```

case2

```
int a=2, b=-3, c=8;   // Polynomial coefficients
float x0 = 18.0;      // Starting point
float lambda = 0.1;   // Learning Rate
```

case3

```
int a=2, b=0, c=8;   // Polynomial coefficients
float x0 = 3.5;      // Starting point
float lambda = 0.1;   // Learning Rate
```

case4

```
int a=4, b=-1, c=0;   // Polynomial coefficients
float x0 = 2.0;      // Starting point
float lambda = 0.1;   // Learning Rate
```