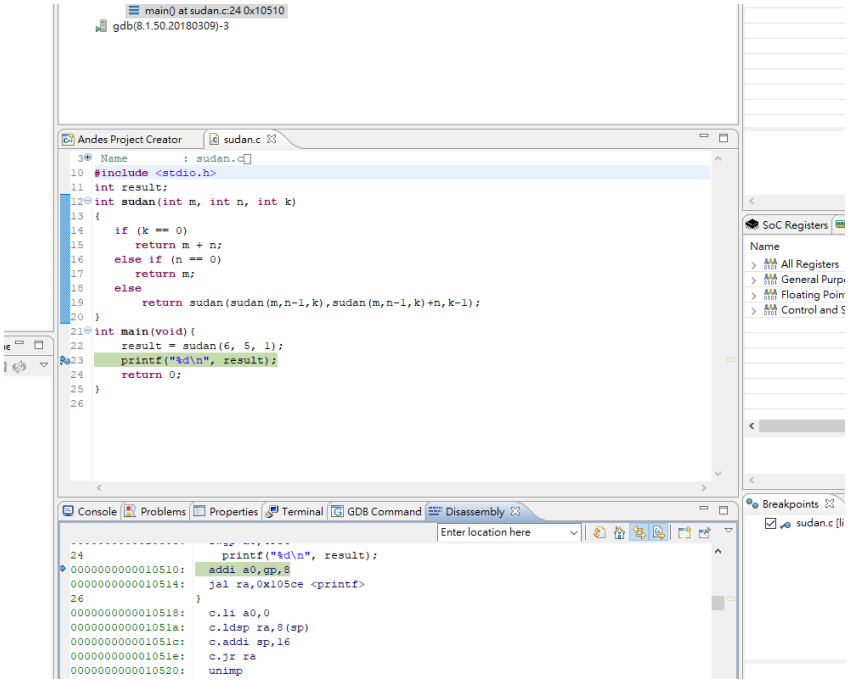
Department of Computer Science
National Tsing Hua University
CS4100 Computer Architecture
Spring 2022 Homework 2
Deadline: 2022/3/31 10:00am

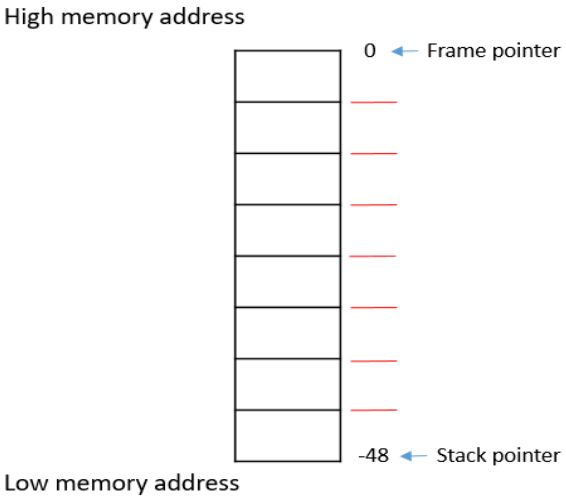
1. (35 points)
   (1) Create an Andes C project and replace it with the downloaded `sudan.c` file.
   (2) Change optimization level to -O0.
   (3) Press the button “Build" in the toolbar.
   (4) Press the button “Debug” in the toolbar.
   (5) Open the Disassembly window.



Note: When you check the assembly code of this project, you may see some instructions that start with a “c.”. These instructions are *RISC-V standard compressed instruction set extension*. The extension reduces program code size by adding short 16-bit instruction encodings for common operations. For example, the instruction `c.add rd,rs2` adds the values in registers `rd` and `rs2` and writes the result to register `rd`. It expands into `add rd,rd,rs2` for normal 32-bit RISC-V instruction encoding. Since there are only two registers in `c.add rd,rs2` (2-address), it is possible to encode the instruction with 16 bits, thereby reducing the code size. You can read the “RISC-V_C_Extension_Instruction_Set.pdf” for more details.
Answer the following questions.

(a) (10 points) First, show how the memory address of the gp register is initialized. Next, show how to get the memory address of the “**result**” variable by referencing the gp register in “main”. Finally, find the memory address of the procedure “**sudan**”.

(b) (10 points)
   i. Record the assembly code from memory address 0x104f4 to 0x1050c. Fill in the stack block by register name and the corresponding memory offsets of the stack in the below figure.
   ii. During the execution of the sudan function, what is the lowest memory address that the stack pointer pointed to?



(c) (5 points) Record the assembly code from memory address 0x104f4 to 0x1050c, which correspond to part of the statement: `return sudan(sudan(m,n-1,k),sudan(m,n-1,k)+n,k-1);` Briefly explain what these instructions do. (You can use a screenshot and explain each line correspondingly.)

(d) (10 points)

i. Change the optimization level to -Og, and do the same as (c) but with the whole statement: `return sudan(sudan(m,n-1,k),sudan(m,n-1,k)+n,k -1).`

ii. Compare the difference of the assembly codes according to the return statements that are generated by the different optimization levels -Og and -O0.

2. (5 points) Respectively show how the value $14A7CF9E_{hex}$ would be arranged in the memory of a little-endian machine and a big-endian machine. Assume that the machines are byte-addressable and the data are stored starting at address 0x00000000.

3. (10 points) For each of the following C statements, write the corresponding RISC-V assembly code. Assume that the base addresses of arrays A and B are in registers x10 and x11, respectively. Each element of A or B is 8 bytes, and the variables g, h, i, j are assigned to registers x5, x6, x7 and x8, respectively.

(a) (5 points) `B[8] = A[g + h];`

(b) (5 points) `i = A[B[4]] - j;`

4. (10 points) Consider the following code sequence, assuming that LOOP is at memory location 1024_{10}. What is the binary representation for the 4th instruction (`beq`) and the 8th instruction (`jal`)?

```
LOOP:  slli x10,x22,3
       add  x10,x10,x1
       ld   x9,0(x10)
       beq  x9,x24,EXIT
       addi x22,x21,2
       addi x22,x22,1
       addi x23,x23,1
       jal  x0,LOOP
EXIT:
```

5. (10 points) (show your derivation)

(a) Give the instruction type and assembly instruction for the following RISC-V machine instruction:
0100 0001 0111 1001 0101 0100 0011 0011

(b) Give the instruction type and hexadecimal representation of the following RISC-V assembly

instruction:

```
sd  x6,  80(x26)
```

6. (10 points) Translate the following RISC-V code into C code. Assume that the C-code integer `result` is held in register `x5`, `x6` holds the C-code integer `i`, and `x10` holds the base address of the integer array `MemArray`.

```
      addi x6, x0, 100
      addi x29, x0, 0
LOOP: ld x7, 0(x10)
      add x5, x5, x7
      addi x10, x10, 8
      addi x6, x6, -4
      bgt x6, x29, LOOP
```

7. (10 points) Implement the following C code in RISC-V assembly. Note: According to RISC-V spec, "In the standard RISC-V calling convention, the stack grows downward and the stack pointer is always kept 16-byte aligned.".

```
int fib(int n){
     if(n == 0)
          return 0;
     else if(n == 1)
          return 1;
     else
          return fib(n-1) + 2*fib(n-2);
}
```

8. (10 points) We would like to expand the RISC-V register file to 128 registers and expand the instruction set to contain four times as many instructions.
   (a) How would this affect the size of each of the bit fields in the R-type instructions?
   (b) How could each of the two proposed changes along decrease the size of a RISC-V assembly program? On the other hand, how could the two proposed changes together increase the size of an RISC-V assembly program?