# Lab 5 Activity

## Understanding Stacks and Procedures

1. Discuss and write the C code for the code below

01 .include beta.uasm

02

03 CMOVE(4, R1)

04 PUSH(R1)

05 BR(square,LP)

06 DEALLOCATE(1)

07 HALT()

08

09 square: PUSH(LP)

10 PUSH(BP)

11 MOVE(SP,BP)

12

13 PUSH(R2)

14 LD(BP,-12,R2)

15 MUL(R2,R2,R0)

16 POP(R2)

17

18 MOVE(BP,SP)

19 POP(BP)

20 POP(LP)

21 JMP(LP)

C – code:

int i = 4;

int square(int a) {

int b;

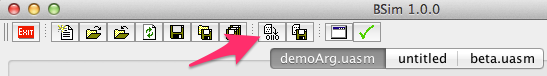
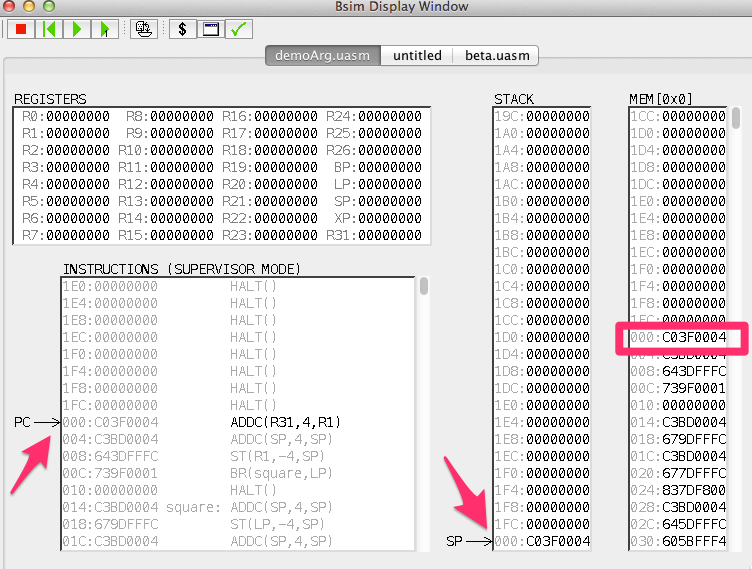
b = a \* a;

return b

}

is = square(i);

The above code is a function call to calculate a square of an integer number.

1. Type the above code in BSim. Modify the path in the .include to the location of your beta.uasm (is provided under 50.002 package file). Note: Do not type in the line numbers.
2. Save the program under BSim, and click “RUN UASM” button.
3. You should see another screen pop out

Write the value of the following:

PC Address: 0x000

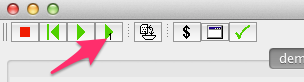
Instruction to be executed: ADDC(R31, 4, R1)

SP Address: 0x000

Get the most significant 6-bit from those data, and compare with the Beta instruction’s opcode. Fill in the table.

|  |  |  |  |
| --- | --- | --- | --- |
| Memory Location | Data in HEX | Most significant 6-bit | Opcode for |
| 0x000: | 0xC03F0004 | 110000 | ADDC |
| 0x004: | 0xC3BD0004 | 110000 | ADDC |
| 0x008: | 0x643DFFFC | 011001 | ST |
| 0x00C: | 0x779F0002 | 011101 | BEQ |
| 0x010: | 0xC7BD0004 | 110001 | SUBC |

1. For the following, you will need to click the “Step” button.



* What is the value of R1 after executing line 03 (CMOVE(4, R1))?

0x4

* What is the memory address of SP pointing to after executing line 04 (PUSH(R1))?

0x004

* At which memory address was R1 pushed into after executing line 04 (PUSH(R1))?

0x000

* What are the values of the following register after executing line 05 (BR(square,LP))?
  + PC : 0x018
  + LP : 0x80000010
* To which instruction does LP point to after executing line 05 (BR(square,LP))?

SUBC(SP, 4, SP)

* What is the address of **square** label on line 09? 0x018
* Fill in the values of the following after executing line 09 (PUSH(LP)):
  + Address that SP points to: 0x008
  + Value of top of the stack : 0x80000010
* Fill in the values of the following after executing line 10 (PUSH(BP)):
  + Address that SP points to: 0x00C
  + Value of top of the stack : 0x0
* What is the value of BP after executing line 11 (MOVE(SP,BP))?

0xC

* Fill in the values of the following after executing line 13 (PUSH(R2)):
  + Address that SP points to: 0x010
  + Value of top of the stack: 0x0
* What is the value of R2 after executing line 14 (LD(BP,-12,R2))?

0x4

* What is the value of R0 after executing line 15 (MUL(R2,R2,R0))?

0x10

* Fill in the values of the following after executing line 16 (POP(R2)):
  + Address that SP points to: 0x00C
  + Value of top of the stack: 0x0
  + Value of R2: 0x0
* Fill in the values of the following after executing line 18 (MOVE(BP,SP)):
  + Address that SP points to: 0x00C
  + Value of top of the stack: 0x0
* Fill in the values of the following after executing line 19 (POP(BP)):
  + Address that SP points to: 0x008
  + Value of top of the stack: 0x80000010
  + Value of BP: 0x0
* Fill in the values of the following after executing line 20 (POP(LP)):
  + Address that SP points to: 0x004
  + Value of top of the stack: 0x4
  + Value of LP: 0x80000010
* Fill in the values of the following after executing line 21 (JMP(LP)):
  + Address that SP points to: 0x004
  + Value of top of the stack: 0x4
  + Address that PC points to: 0x010
  + Value of R0: 0x10
  + Value of R1: 0x4
  + Value of R2: 0x0

1. What is the data contained in the following memory locations at the end of all execution?

* 0x000 : 0x4
* 0x004 : 0x80000010
* 0x008 : 0x0
* 0x00C : 0x0
* 0x010 : 0xC7BD0004

1. Compare your result in Step 6 to the one in Step 4. Discuss the problem of the current code in relation to the stack implementation.
2. Propose a solution to the problem and try it out on BSim.

## Conditionals

1. Complete the assembly language code below to correspond to its C-code.

|  |  |
| --- | --- |
| .include beta.uasm  CMOVE(2, R1)  CMOVE(4, R2)  CMOVE(0, R3)  CMPLT(R1, R2, R4)  BEQ(R4, if\_true)  MOVE(R2, R3)  BR(done)  if\_true:  MOVE(R1, R3)  done:  ADDC(R3, 1, R3)  HALT() | int x = 2;  int y = 4;  int z = 0;  if (x >= y)  z = x;  else  z = y;  z = z+1; |

1. Check your assembly code by running in BSim step by step.

## Loops

1. Complete the assembly language code below to correspond to its C-code.

|  |  |
| --- | --- |
| .include beta.uasm  CMOVE(5, R1)  CMOVE(0, R2)  CMOVE(0, R3)  CMPLT(R3, R1, R4)  BNE(R4, loop)  loop:  ADD(R2, R3, R2)  ADDC(R3, 1, R3)  CMPLT(R3, R1, R4)  BNE(R4, loop)  HALT() | int n = 5;  int sum = 0;  for (int i = 0; i<n ; i++)  sum+=i; |

1. Check your assembly code by running in BSim step by step.