Lab 5 Activity

Understanding Stacks and Procedures

1. Discuss and write the C code for the code below
   Caller – 3 to 8, Callee – lines 9 onwards

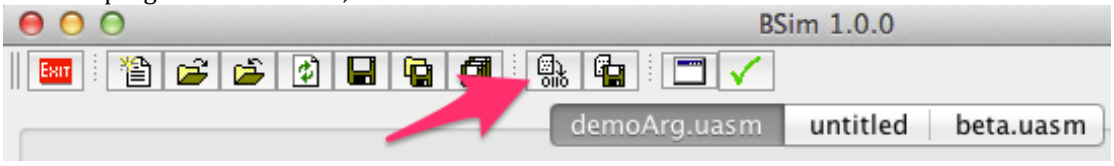
```
01 .include beta.uasm
02
03      CMOVE(4, R1)

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:

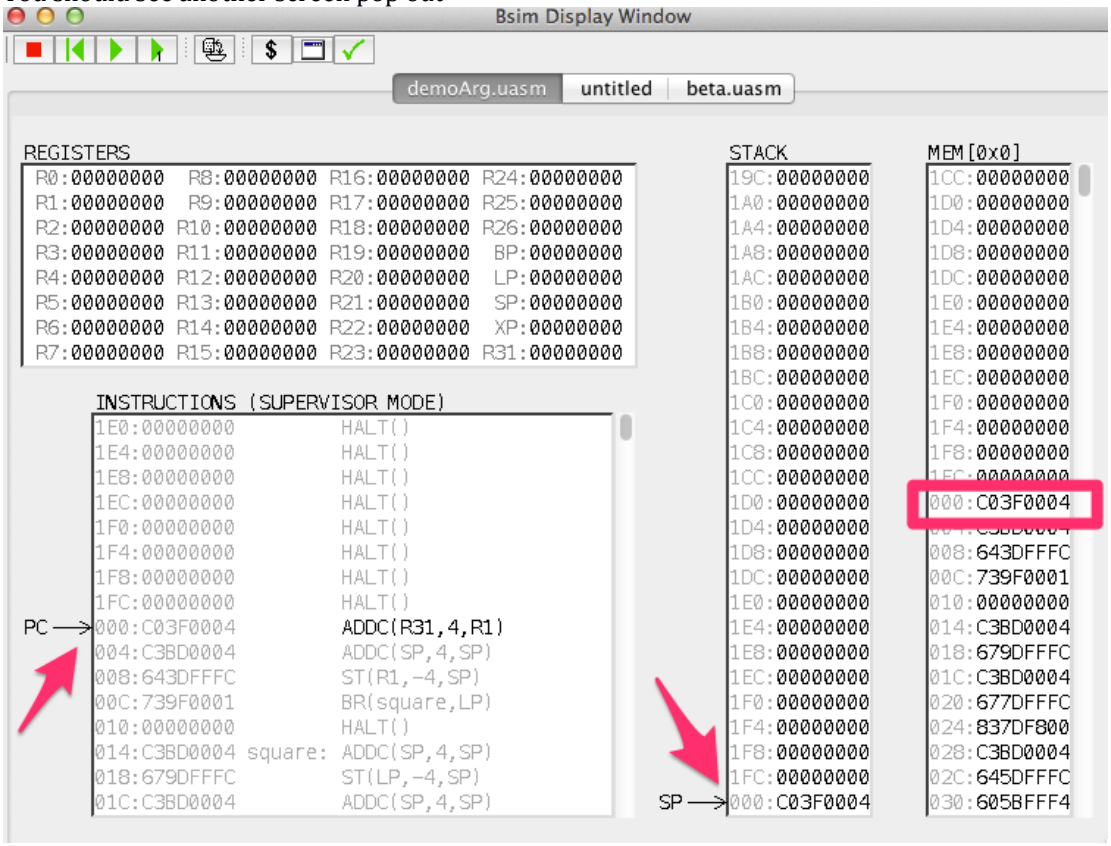
The above code is a function call to calculate a __square__ of an integer number.

2. 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.

3. Save the program under BSim, and click "RUN UASM" button.



4. You should see another screen pop out



Write the value of the following:

PC Address: __000__

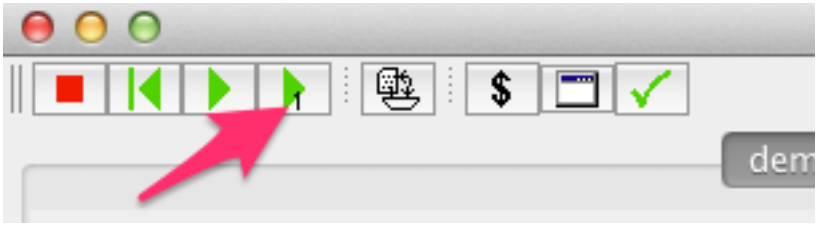
Instruction to be executed: __ADDC(R31,4,R1)_____

SP Address: __C03F004__

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:	C03F004	110000	ADDC
0x004:	C3BD004	110000	ADDC
0x008:	643DFFFC	011001	ST
0x00C:	739F0001	011100	BR
0x010:	000000	000000	-

5. For the following, you will need to click the "Step" button.



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

  __4___

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

  __0000 0004__

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

  ____000___

- What are the values of the following register after executing line 05 (BR(square,LP))?

  - PC : ___C3BD0004_____
  - LP : ____8000 0010____

- To which instruction does LP point to after executing line 05 (BR(square,LP))?

  ___ADDC(SP,4,SP)___

- What is the address of **square** label on line 09? ____018_____
- Fill in the values of the following after executing line 09 (PUSH(LP)):

  - Address that SP points to: _____008____
  - Value of top of the stack : ___8000 0010__

- Fill in the values of the following after executing line 10 (PUSH(BP)):

  - Address that SP points to: __00C___
  - Value of top of the stack : ____0000 0000_____

- What is the value of BP after executing line 11 (MOVE(SP,BP))?

  __0000 000C_____

- Fill in the values of the following after executing line 13 (PUSH(R2)):

- Address that SP points to: __010__
- Value of top of the stack: __0000 0000__

- What is the value of R2 after executing line 14 (LD(BP,-12,R2))?

  __4__

- 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: __00C__
  - Value of top of the stack: __0000 0000__
  - Value of R2: __0__
- Fill in the values of the following after executing line 18 (MOVE(BP,SP)):
  - Address that SP points to: __00C__
  - Value of top of the stack: __0000 0000__

- Fill in the values of the following after executing line 19 (POP(BP)):
  - Address that SP points to: __008__
  - Value of top of the stack: __8000 0010__
  - Value of BP: __0000 0000__
- Fill in the values of the following after executing line 20 (POP(LP)):
  - Address that SP points to: __004__
  - Value of top of the stack: __0000 0004__
  - Value of LP: __8000 0010__
- Fill in the values of the following after executing line 21 (JMP(LP)):
  - Address that SP points to: __004__
  - Value of top of the stack: __0000 0004__
  - Address that PC points to: __010__
  - Value of R0: __0x10__
  - Value of R1: __4__
  - Value of R2: __0__

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

   - 0x000 : _____0000 0004_____
   - 0x004 : _____8000 0010_____
   - 0x008 : ___0000 0000____
   - 0x00C : ___0000 0000_____
   - 0x010 : __C7BD 0004_____

7. 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.

Initial code got overwritten. Stack and code pointing at the same place initially at address 000. When you push into stack, you are changing the code.

1. Initialise stack pointer
   at start of code: LDR(StackBase,SP)
   Later: StackBase: LONG(.+4)
   .=.+0x100 || allocate bytes for data

   current address + 16**2 bytes (size of stack)

8. 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,RO)
CMOVE(4,R1)
CMOVE(0,R2)
 	
CMPLT(R0,R1,R3)
BT(R3,L1,R31)
MOVE(R0,R2)
BR(L2,R31) | R31, don needa come back
 	
L1: MOVE(R1,R2)
 	
L2: ADDC(R2,1,R2)
	int x = 2;
	int y = 4;
	int z = 0;
	
	if (x >= y)
	  z = x;
	else
	  z = y;
	
	z = z+1;

2. 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) | n
CMOVE(0,R2) | sum
CMOVE(0,R3) | i
BR(Check)
 	
Body:
  ADD(R2,R3,R2) |sum=sum+i
  ADDC(R3,1,R3) |i++
 	
Check:
  CMPLT(R3,R1,R0)
  BT(R0,body,R31)
  | don't wanna keep the location if not true
	int n = 5;
	int sum = 0;
	for (int i = 0; i<n ; i++)
	  sum+=i;

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