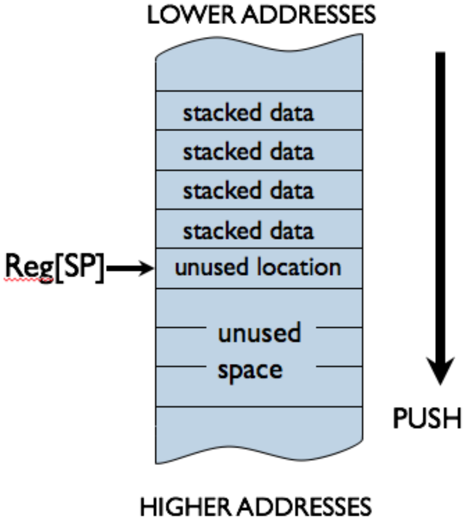
Computation Structures

Procedures & Stacks Worksheet



PUSH(X): Push Reg[x] onto stack

```
  ADDC(SP,4,SP)
  ST(Rx,-4,SP)
```

POP(X): Pop value at top of stack into Reg[x]

```
  LD(SP,-4,RX)
  SUBC(SP,4,SP)
```

ALLOCATE(k): Reserve k words of stack

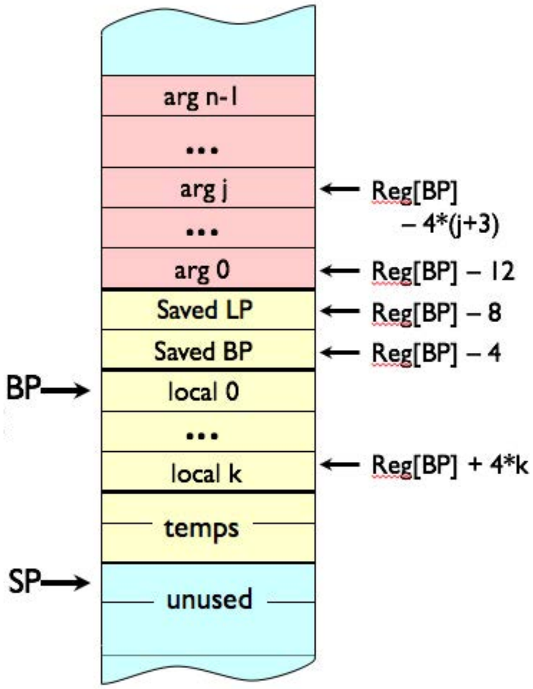
```
  ADDC(SP,4*k,SP)
```

DEALLOCATE(k): Release k words of stack

```
  SUBC(SP,4*k,SP)
```

Stack discipline: leave stack the way you found it => for every PUSH(), there's a corresponding POP() or DEALLOCATE()

Activation record layout on the stack (aka stack frame):



CALLING SEQUENCE

```
   PUSH(argn)    // push args, last arg first
   …
   PUSH(arg1)
   BR(f, LP)     // call f, return addr in LP
   DEALLOCATE(n) // remove args from stack
```

ENTRY SEQUENCE

```
f: PUSH(LP)      // save return addr
   PUSH(BP)      // save old frame pointer
   MOVE(SP,BP)   // initialize new frame pointer
   ALLOCATE(nlocals)  // make room for locals
   (push other regs)  // preserve old reg vals
```

EXIT SEQUENCE

```
   // return value in R0
   MOVE(BP,SP)   // remove locals
   POP(BP)       // restore old frame pointer
   POP(LP)       // recover return address
   JMP(LP)       // resume execution in caller
```

Problem 1.

You are given an incomplete listing of a C program (shown below) and its translation to Beta assembly code (shown on the right):

```
int fn(int x) {
  int lowbit = x & 1;
  int rest = x >> 1;
  if (x == 0) return 0;
  else return ???;
}
```

```
fn: PUSH(LP)
    PUSH(BP)
    MOVE(SP,BP)
    ALLOCATE(2)
    PUSH(R1)
    LD(BP,-12,R0)
    ANDC(R0,1,R1)
xx: ST(R1,0,BP)
    SHRC(R0,1,R1)
    ST(R1,4,BP)
yy: BEQ(R0,rtn)
    LD(BP,4,R1)
    PUSH(R1)
    BR(fn,LP)
    DEALLOCATE(1)
    LD(BP,0,R1)
    ADD(R1,R0,R0)
rtn:POP(R1)
zz: MOVE(BP,SP)
    POP(BP)
    POP(LP)
    JMP(LP)
```

(A) What is the missing C source corresponding to ??? in the above program?

C source code: ____________________________________

(B) Suppose the instruction bearing the tag '**zz:**' were eliminated from the assembly language program. Would the modified procedure work the same as the original procedure (circle one)?

Work the same? YES … NO

(C) In the space below, fill in the binary representation for the instruction stored at the location tagged '**xx:**' in the above program.

(fill in missing 1s and 0s for instruction at xx:)

The procedure **fn** is called from an external procedure and its execution is interrupted just prior to the execution of the instruction tagged '**yy:**'. The contents of a region of memory are shown on the left below.

NB: All addresses and data values are shown in hex. The contents of **BP** are 0x1C8 and **SP** contains 0x1D4.

```
184:   4
188:   7
18C:  47
190:  C4
194: 170
198:   1
19C:  23
1A0:  22
1A4:  23
1A8:  4C
1AC: 198
1B0:   1
1B4:  11
1B8:  23
1BC:  11
1C0:  4C
1C4: 1B0
1C8:   1   ←BP
1CC:   8
1D0: ???
1D4:   0   ←SP
```

(D) What was the argument to the most recent call to **fn**?

Most recent argument (HEX): x=_______

(E) What is the missing value marked **???** for the contents of location 1D0?

Contents of 1D0 (HEX): _______

(F) What is the hex address of the instruction tagged **rtn:?**

Address of rtn (HEX): _______

(G) What was the argument to the *original* call to **fn**?

Original argument (HEX): x=_______

(H) What is the hex address of the BR instruction that called **fn** *originally*?

Address of original call (HEX): _______

(I) What were the contents of R1 at the time of the *original* call?

Original R1 contents (HEX): _______

(J) What value will be returned to the *original* caller?

Return value for original call (HEX): _______

Problem 2.

You are given an incomplete listing of a C program (shown below) and its translation to Beta assembly code (shown on the right):

```
int f(int x, int y) {
  x = (x >> 1) + y;
  if (y == 0) return x;
  else return ???;
}
```

```
f:   PUSH(LP)
     PUSH(BP)
     MOVE(SP,BP)
     PUSH(R1)
     LD(BP,-12,R0)
     SHRC(R0,1,R0)
     LD(BP,-16,R1)
     ADD(R0,R1,R0)
     BEQ(R1,rtn)
     SUBC(R1,1,R1)
     PUSH(R1)
     PUSH(R0)
     BR(f,LP)
     DEALLOCATE(2)
rtn: POP(R1)
zz:  MOVE(BP,SP)
     POP(BP)
     POP(LP)
     JMP(LP)
```

(A) What is the missing C source corresponding to ??? in the above program?

C source code: ________________________________

(B) Suppose the instruction bearing the tag '**zz:**' were eliminated from the assembly language program. Would the modified procedure work the same as the original procedure?

Work the same (circle one)? YES … NO

The procedure **f** is called from an external procedure and then execution is stopped just prior to one of the executions of the instruction labeled '**rtn:**'. The addresses and contents of a region of memory are shown in the table on the right; all addresses and data values in the table are in hex. When execution is stopped **BP contains the value 0x14C and SP contains the value 0x150.**

108	7
10C	320
110	104
114	3
118	A
11C	2C4
120	104
124	3
128	2
12C	
130	348
134	124
138	2
13C	1
140	6
144	348
148	138
14C	1
150	0
154	4
158	348
15C	14C
160	0

(C) What are the arguments to the **currently active call** to **f**?

Most recent arguments (in hex): x = 0x_______, y = 0x_______

(D) If you can tell from the information provided, specify the arguments to the **original** call to **f**, otherwise select **CAN'T TELL**.

Original arguments (in hex) : x = 0x_____, y = 0x_____, or CAN'T TELL

(E) What is the missing value in location 0x12C?

Contents of location 0x12C (in hex): 0x_______

(F) What is the hex address of the instruction labeled **rtn:?**

Address of instruction labeled rtn: (in hex): 0x_______

(G) What is the hex address of the BR instruction that called **f** *originally*?

Address of original call (in hex): 0x_______, or CAN'T TELL

(H) What value will be returned to the *original* caller?

Return value for original call (in hex): 0x_______

Problem 3.

The following C program implements a function H(x,y) of two arguments, which returns an integer result. The assembly code for the procedure is shown on the right.

```
int H(int x, int y) {
    int a = x - y;
    if (a < 0) return x;
    else return ???;
}
```

```
H:     PUSH(LP)
       PUSH(BP)
       MOVE(SP, BP)
       ALLOCATE(1)
       PUSH(R1)

       LD(BP,-12,R0)
       LD(BP,-16,R1)
       SUB(R0,R1,R1)
       ST(R1,0,BP)

       CMPLTC(R1,0,R1)
       BT(R1,rtn)

       LD(BP,-16,R1)
       PUSH(R1)
       LD(BP,0,R0)
       PUSH(R0)
       BR(H,LP)
       DEALLOCATE(2)

rtn:   POP(R1)
       MOVE(BP,SP)
       POP(BP)
       POP(LP)
       JMP(LP)
```

The execution of the procedure call **H(0x68,0x20)** has been suspended just as the Beta is about to execute the instruction labeled "rtn:" during one of the recursive calls to H. A *partial* trace of the stack at the time execution was suspended is shown to the right below.

(A) Examining the assembly language for H, what is the appropriate C code for ??? in the C representation for H?

C code for ???: ________________________________

(B) Please fill in the values for the blank locations in the stack dump shown on the right. Express the values in hex or write "---" if value can't be determined. Hint: Figure out the layout of H's activation record and use it to identify and label the stack frames in the stack dump.

Fill in the blank locations with values (in hex!) or "---"

(C) Determine the specified values at the time execution was suspended. Please express each value in hex or write "CAN'T TELL" if the value cannot be determined.

Value in R0 or "CANT TELL": 0x_______________

Value in R1 or "CANT TELL": 0x_______________

Value in BP or "CANT TELL": 0x_______________

Value in LP or "CANT TELL": 0x_______________

Value in SP or "CANT TELL": 0x_______________

	0x0024
	0x0070
	0x0048
	0x0068
	0x0020
	0x0020
	0x0028
	0x007C
	0x00C8
BP→	0x0008
	0x0020
	0x0020

Problem 4.

The following C program computes the log base 2 of its argument. The assembly code for the procedure is shown on the right, along with a stack trace showing the execution of ilog2(10). The execution has been halted just as it's about to execute the instruction labeled "rtn:"

```
/* compute log base 2 of arg */
int ilog2(unsigned x) {
    unsigned y;
    if (x == 0) return 0;
    else {
        /* shift x right by 1 bit */
        y = x >> 1;
        return ilog2(y) + 1;
    }
}
```

```
ilog2: PUSH(LP)
       PUSH(BP)
       MOVE(SP,BP)
       ALLOCATE(1)
       PUSH(R1)

       LD(BP,-12,R0)
       BEQ(R0,rtn,R31)

       LD(BP,-12,R1)
       SHRC(R1,1,R1)
       ST(R1,0,BP)

       LD(BP,0,R1)
       PUSH(R1)
       BR(ilog2,LP)
       DEALLOCATE(1)
       ADDC(R0,1,R0)

rtn:   POP(R1)
xxx:   DEALLOCATE(1)
       MOVE(BP,SP)
       POP(BP)
       POP(LP)
       JMP(LP)
```

(A) What are the values in R0, SP, BP and LP at the time execution was halted? Please express the values in hex or write "CAN'T TELL".

Value in R0: 0x_______________ **in SP: 0x**_______________

Value in BP: 0x_______________ **in LP: 0x**_______________

(B) Please fill in the values for the five blank locations in the stack trace shown on the right. Please express the values in hex.

Fill in values (in hex!) for 5 blank locations

(C) In the assembly language code for ilog2 there is the instruction "LD(BP,-12,R0)". If this instruction were rewritten as "LD(SP,NNN,R0)" what is correct value to use for NNN?

Correct value for NNN: _______________

(D) In the assembly language code for ilog2, what is the address of the memory location labeled "xxx:"? Please express the value in hex.

Address of location labeled "xxx:": 0x_______________

Values are in hex!

	5
	1A8
	208
	2
	5
	1
	1A8
	230
BP→	0
	1
	0