CS333: Computer Architecture
University of Virginia Computer Science
Fall 2006 — Michele Co

Class 7:
Intro to SRC Simulator
Procedure Calls
HLL -> Assembly

Announcements

- Homework #2
  - Due next Wednesday, Sept. 13 in class
- Class notes posted on course website

Agenda

- SRC and Procedure Calls
  - Calling convention discussed here will be used for SRC programming lab
- Sample problems from Chapter 2

SRC – Procedure Calls

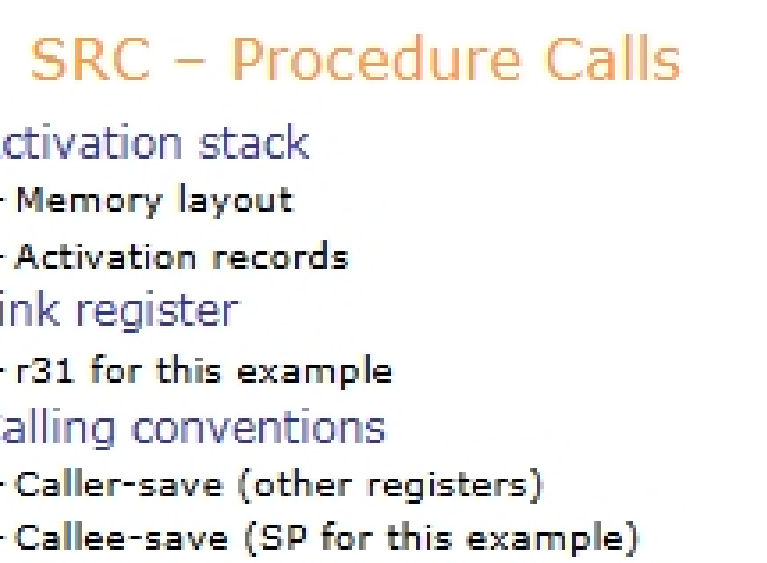
- Activation stack
  - Memory layout
  - Activation records
- Link register
  - r31 for this example
- Calling conventions
  - Caller-save (other registers)
  - Callee-save (SP for this example)

High Level Program

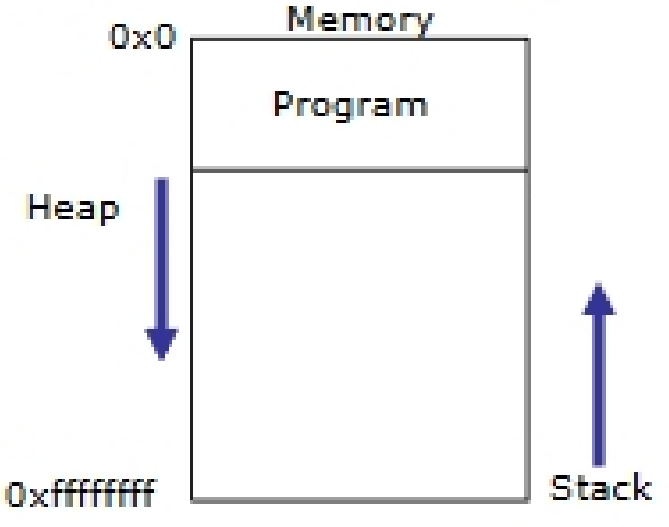
```
int arg1 = -5;    // argument to pass
int output = 0;   // result

int absolute(int param1){
  if(param1 <= 0)
     param1 = -param1;
}
void main(){
  output = absolute(arg1);
}
```

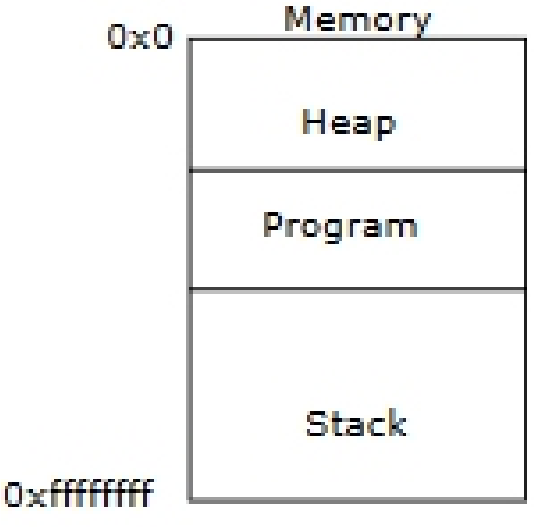

SRC Memory Layout



Typical Memory Layout



Why Wouldn't This Layout Be As Good?



Activation Stack

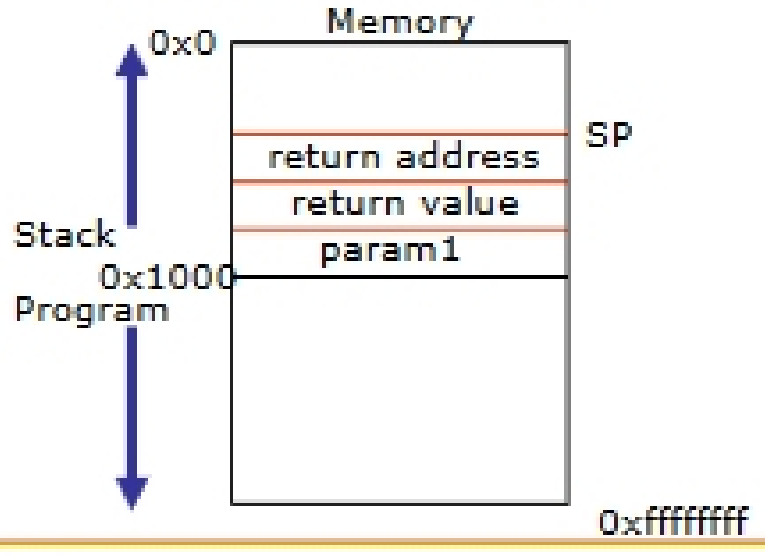
- Stack
  - Last In First Out (LIFO)
    - Push, pop operations
  - Can be located anywhere in memory
    - Typically started in high memory
    - For SRC, we'll use a different memory layout

What is Saved on the Stack?

- Activation Records
  - Return address
  - Function parameters
  - Local variables
  - Space for return values

Information associated with a procedure call

Activation Record



Caller vs. Callee

- Caller
  - Calling function
- Callee
  - Function being called

High Level Program

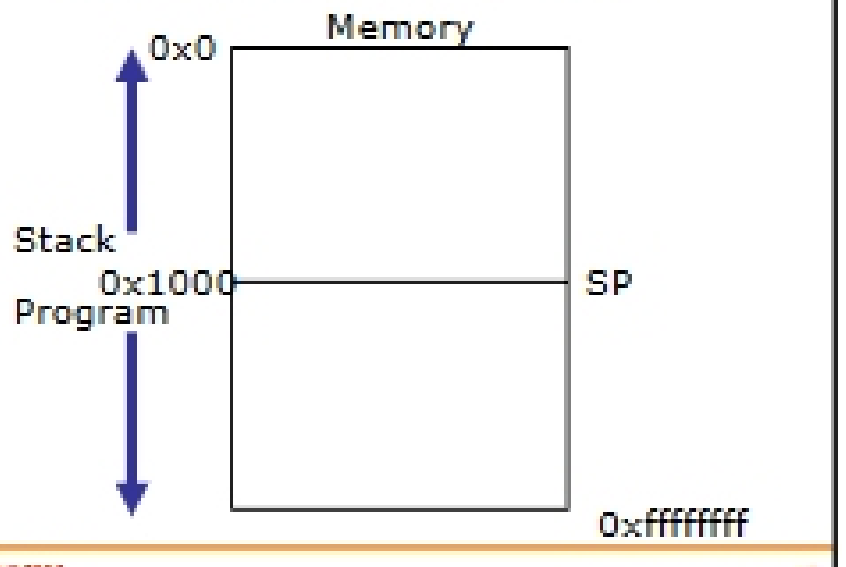
```
int arg1 = -5;    // argument to pass
int output = 0;   // result

int absolute(int param1){
  if(param1 <= 0)
     param1 = -param1;
  return param1;
}
void main(){
  output = absolute(arg1);
}
```

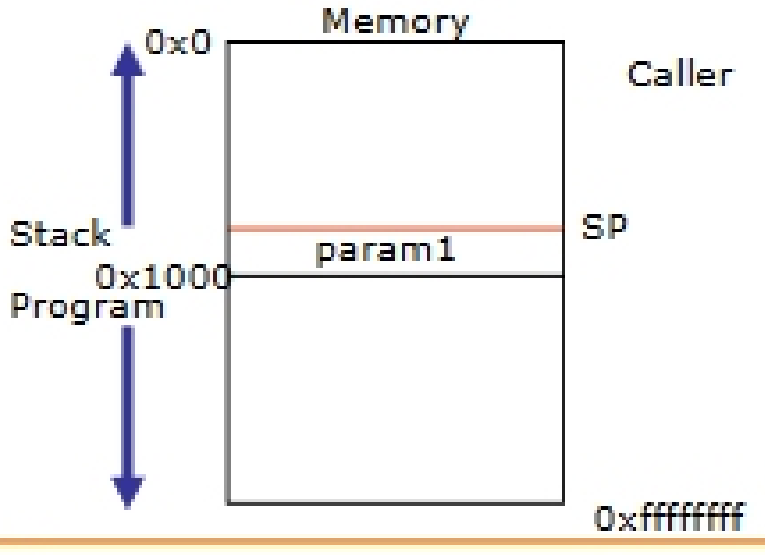

How is a Procedure Call Performed?

- Caller
  1. Stores arguments onto stack
  2. Allocates space for output values (return value)
  3. Calls the procedure
- Callee
  1. Stores SP
  2. Stores the return address
  3. Retrieves any passed arguments
  4. Performs procedure body
  5. Saves results
  6. Returns to caller

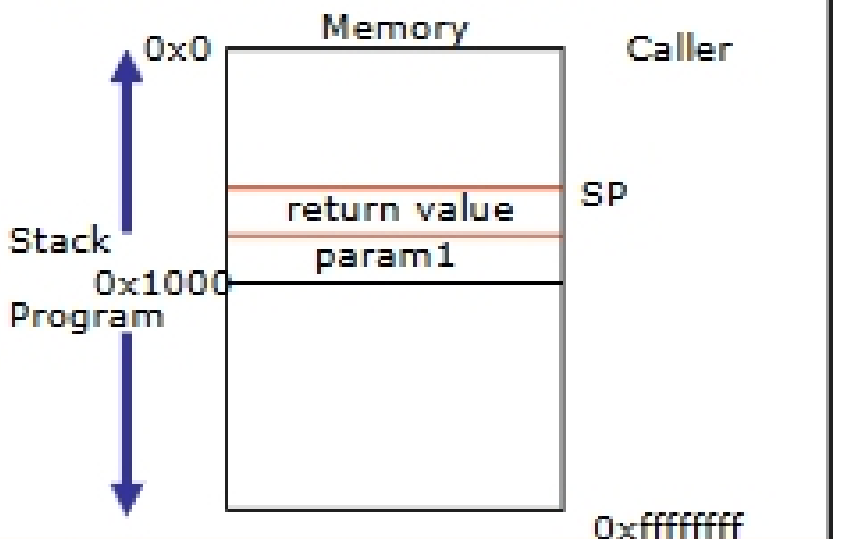
procedure_call.asm



procedure_call.asm



procedure_call.asm



procedure_call.asm