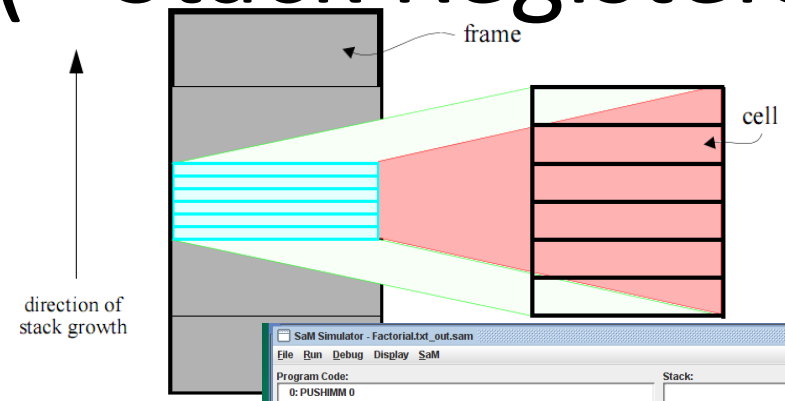
SaM Functions

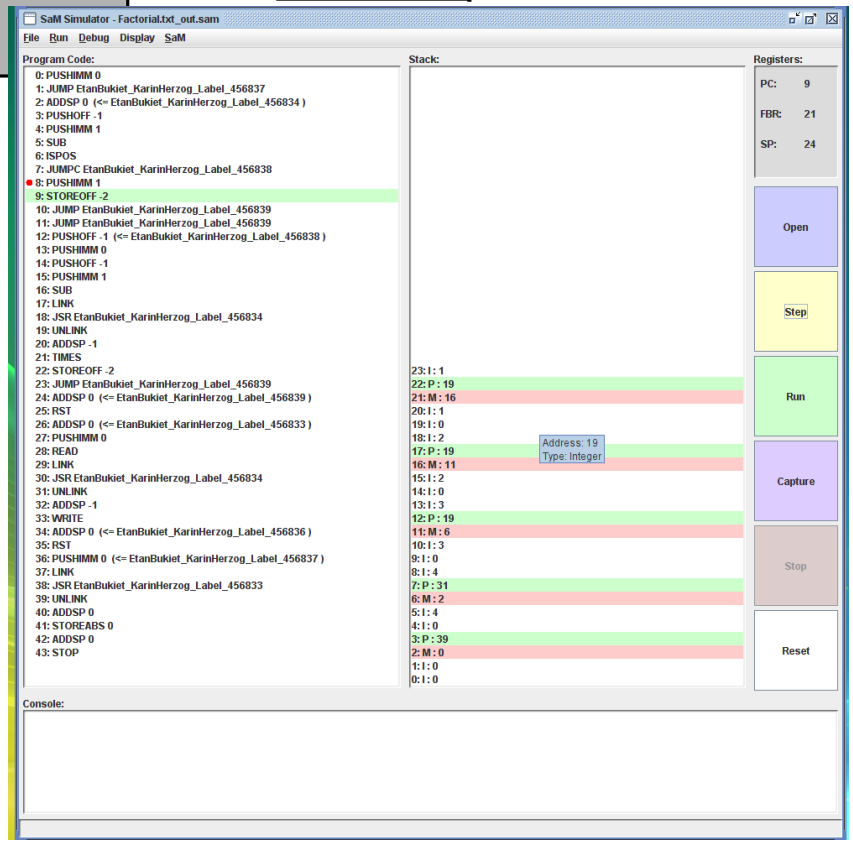
About Functions

- Purpose: Encapsulate procedural information
- Control flow and call order
  - Callee vs. Caller
- Scope (Global vs. Local)
- Parameters
- Recursion
  - Each function call as a separate invocation

The Stack (+ Stack Registers)

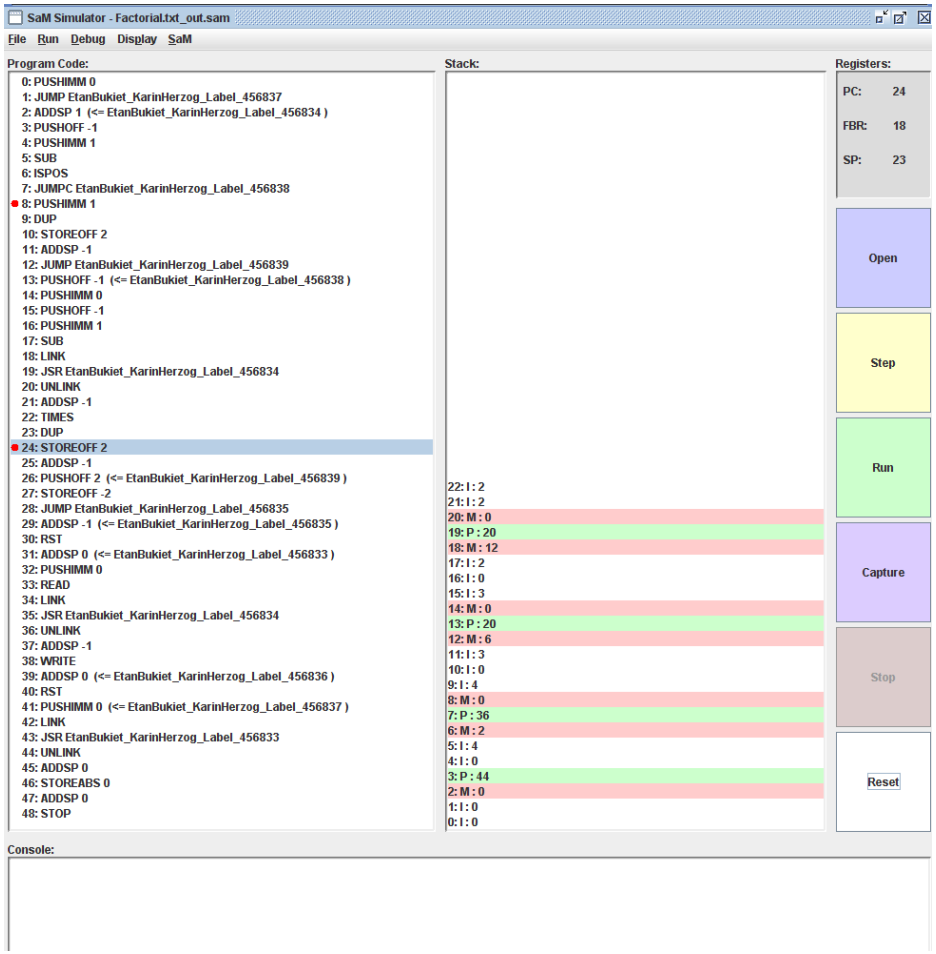
- Stack Registers
  - Frame Base Register
  - Stack Point
  - Program Counter
- Functional Frames
  - Return Value
  - Parameters
  - Return Frame
  - Return Address
  - Local Variables
  - (Eventually: “this” pointer)





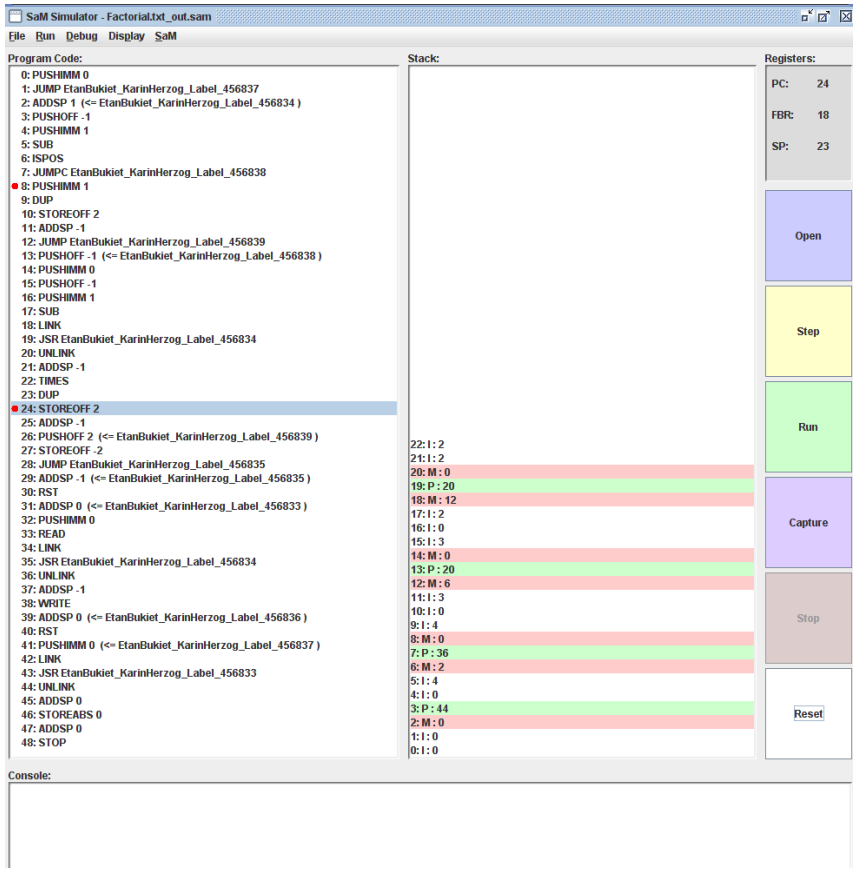
Program Counter, Program Addresses

- Program Counter
- Labels
- Breakpoints



Function Calling Convention

- Caller:
  // Push the function address
  // push arguments
  LINK
  PUSHSP
  PUSHIMM (2+ArgumentsSize)
  SUB
  PUSHIND
  JSIND
  UNLINK
  ADDSP -n // remove params
- Callee:
  function_label:
  ADDSP lv // local vars
  ADDSP -lv
  RST



What does the stack look like?

1

Function Address (also, return var slot)

3

Function Address
OLD FBR
(Arguments)
Function Address (also, return var slot)

2

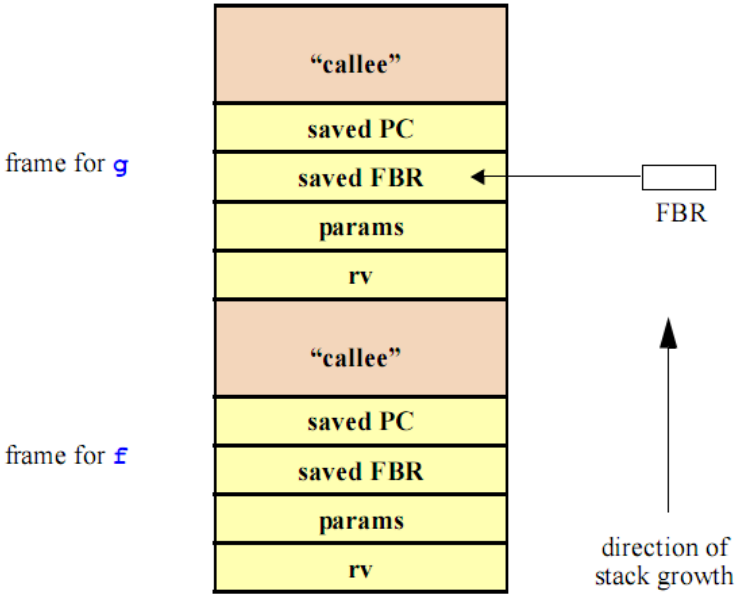
(Arguments)
Function Address (also, return var slot)

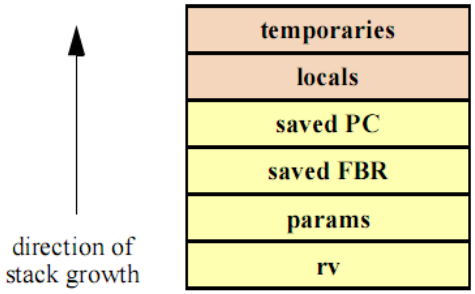
4

Function Address
OLD FBR
(Arguments)
Function Address (also, return var slot)

JSR and RST ?

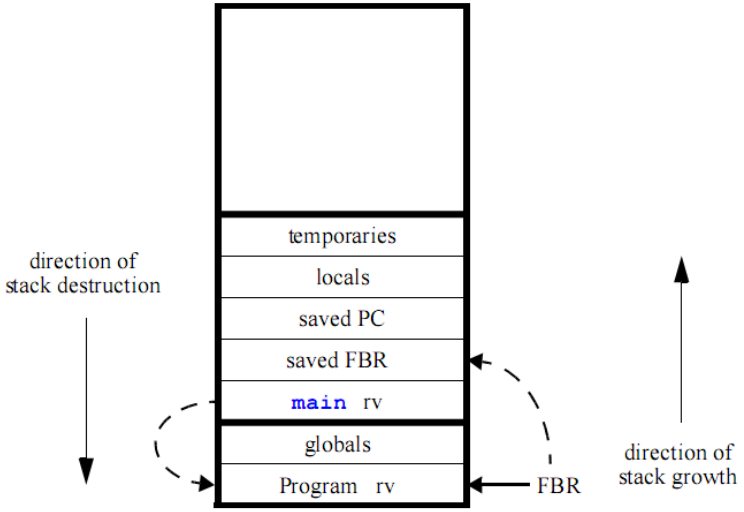
- JSR ?
  - Stack[SP] <= PC+1
  - SP++
  - PC <= label
- RST ?
  - PC <= Stack[SP]
  - SP--





“Program” a wrapper for main?

- The FBR of “Program” is address 0
- The return variable of “Program” is address 0
- The variables of “Program” are any global variables



Returning from a method?

- If returning from a return statement:
  - Evaluate return expression
  - Store result in the return variable slot
  - De allocate local variables that have been allocated until this point
  - Jump to return from method
- At end of method
  - Block cleans up local variables associated with it
  - Write default return value to stack
  - Store result in return variable slot
  - Return-from-method label:
  - Issue RST instruction

Code Walkthrough (Bali)

```
class Entry
{
        public int fact(int i)
        {
                if(i <= 1) { return 1; }
                else { return i*fact(i-1); }
        }
        public int main()
        {
                print(fact(5));
        }
}
```

Code Walkthrough (SaM)

```
PUSHIMM 0                    // allocate the program return variable
PUSHIMMPA UniqueLabel_3      // push the address of Entry::main
LINK                         // copies curr FBR to stack and assigns FBR = SP
PUSHSP                       // calculate the address of the
PUSHIMM 2                    //          function's address
SUB                          //          that we already put on the stack
PUSHIND                      //          and retrieve it
JSRIND                       // and then jump to it
UNLINK                       // when we return, put the FBR back
STOREABS 0                   // put the return value of Entry::main into the
STOP                         //          program return slot, and STOP
UniqueLabel_1:               // Function  Entry::fact
PUSHOFF -1                   // Push i (1'st argument) to the top of the stack
PUSHIMM 1                    // Push the constant 1
GREATER                      // See if i <= 1
NOT
NOT                          // if it's not
JUMPC UniqueLabel_6          //          then skip the if statement
PUSHIMM 1                    // Push the constant we're returning
STOREOFF -2                  //          and store it in the return variable slot
JUMP UniqueLabel_2           //          and jump to the End of function label
```

Code Walkthrough (cont…)

```
UniqueLabel_6:              // otherwise (i > 1)
PUSHOFF -1                  // Push i
PUSHIMMPA UniqueLabel_1     // The we need to call
PUSHOFF -1                  // Push i
PUSHIMM 1                   // Push -1
SUB                         // = i-1 (also, argument 0 to Entry::fact)
LINK                        // copies curr FBR to stack and assigns FBR = SP
PUSHSP                      // calculate the address of the
PUSHIMM 3                   //          function's address
SUB                         //          that we already put on the stack
PUSHIND                     //          and retrieve it
JSRIND                      // and then jump to it
UNLINK                      // when we return, put the FBR back
ADDSP -1                    //  remove the arguments (only 1)
TIMES                       // i*fact(i-1)
STOREOFF -2                 // set the return variable
JUMP UniqueLabel_2          // and jump to the end-of-function label
PUSHIMM 0                   // push the default return value
STOREOFF -2                 //          and store it in the return value slot
UniqueLabel_2:              // End of function label
RST                         // End function: Entry::main
```

Code Walkthrough (cont…)

```
UniqueLabel_3:                  // Function  Entry::main
PUSHIMMPA UniqueLabel_1         // push address of Function Entry::fact
PUSHIMM 5                       // push the argument (=5)
LINK                            // copies curr FBR to stack and assigns FBR = SP
PUSHSP                          // calculate the address of the
PUSHIMM 3                       //          function's address
SUB                             //          that we already put on the stack
PUSHIND                         //          and retrieve it
JSRIND                          // and then jump to it
UNLINK                          // when we return, put the FBR back
ADDSP -1                        // remove the single argument
WRITE                           // print the result
PUSHIMM 1                       // the print expression returns 1
ADDSP -1                        // since it's a statement, we need to remove that value
PUSHIMM 0                       // our function has no return statement, so push default
STOREOFF -1                     //          return value and store it in the return slot
UniqueLabel_4:                  // End of function label
RST                             // End function: Entry::main
```

Code Example 2 (Code)

```
class Entry
{
        int GCF(int x, int y)
        {
                if ((x % y) == 0) { return y; }
                else { return GCF(y, x % y); }
        }
        public int main()
        {
                return GCF(543278, 5436);
        }
}
```