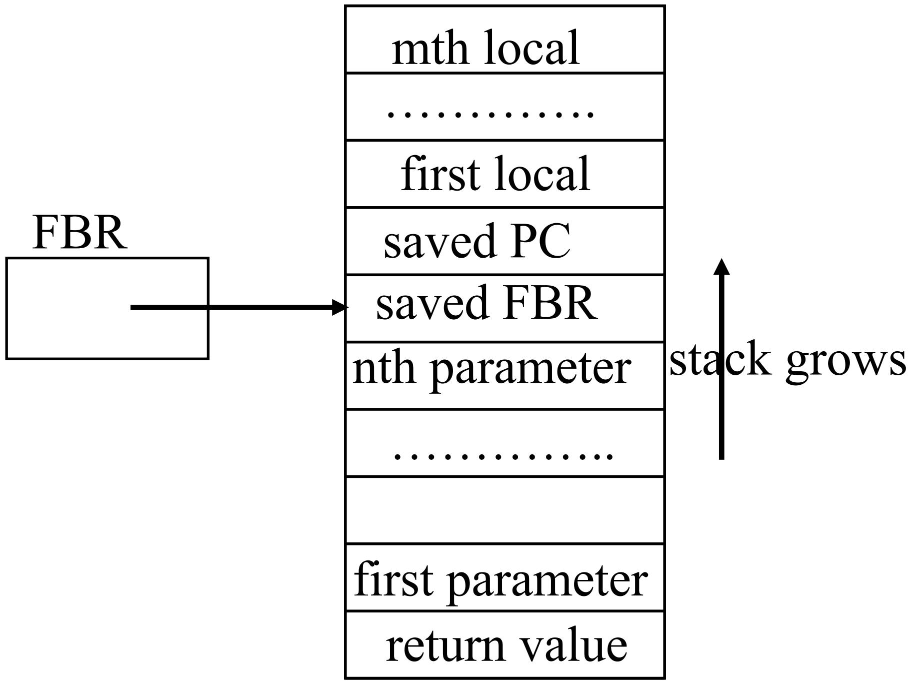
Method Constructs

- Three source-language constructs related to methods.
  1. Invocation: `f(e1, ..., en)`
  2. Definition: `f(p1, ..., pn) {int x1, ..., xc; B}`
  3. Return: `return e;`

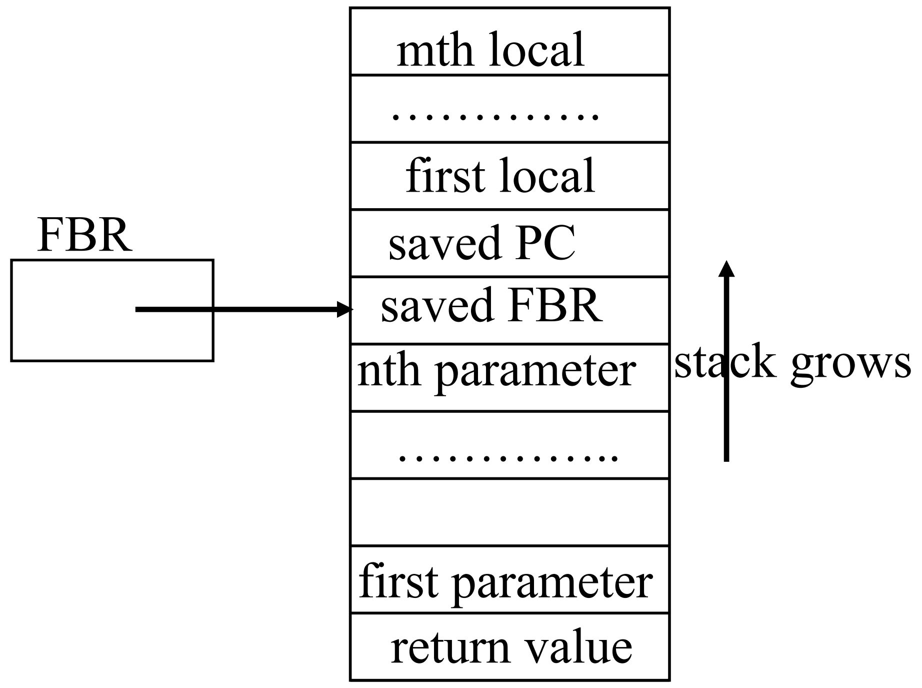
Method Constructs

- Three source-language constructs related to methods.
  1. Invocation: `f(e1, ..., en)`
  2. Definition: `f(p1, ..., pn) {int x1, ..., xc; B}`
  3. Return: `return e;`



Code Generation for Method Invocation

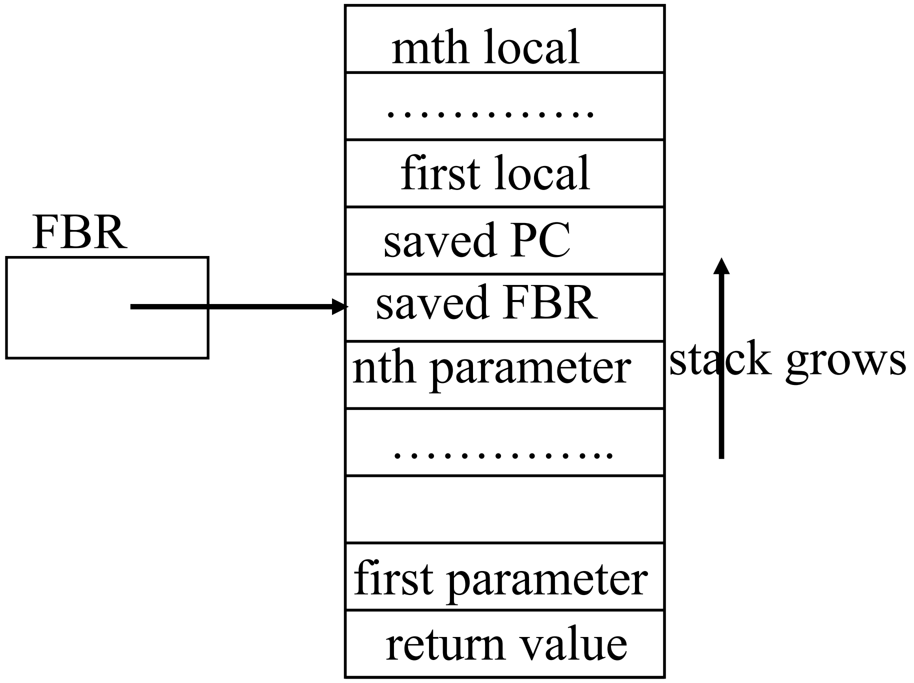
```
f(e1, ..., en)
```



```
PUSHIMM 0  //return value slot
Code for e1
...
Code for en
LINK  //save FBR and update it
JSR f
POPFBR  //restore FBR
ADDSP –n  //pop parameters
```

Code Generation for Method Definition

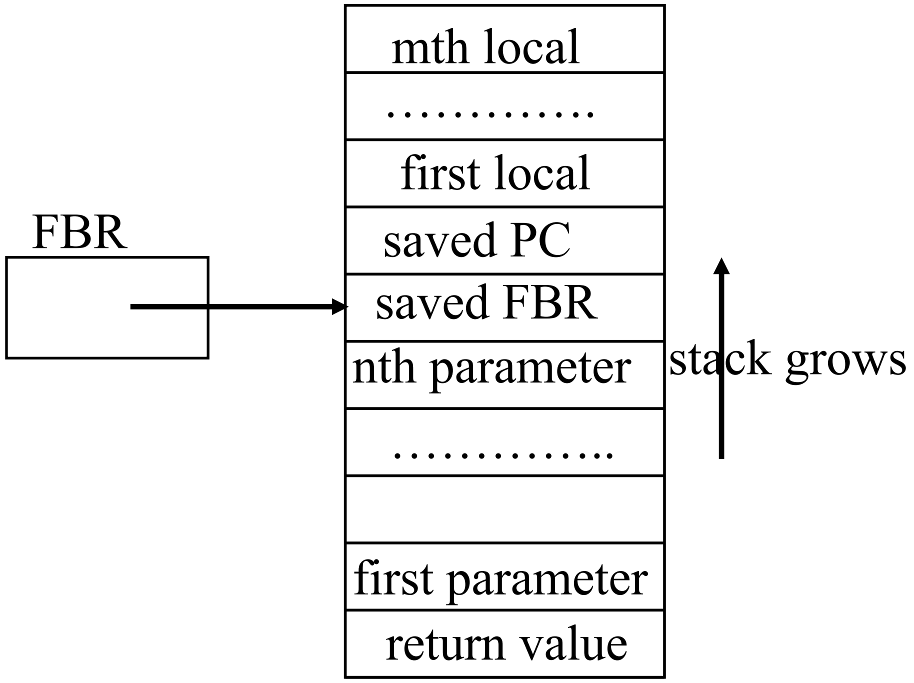
```
f(p1, ..., pn) {int x1, ..., xc; B}
```



```
  ADDSP c   // c is number of locals
  Code for B
fEnd:
  STOREOFF r  //r is offset of rv slot
  ADDSP –c  //pop locals off
  JUMPIND  //return to caller
```

Code Generation for Return

```
return e;
```



Code for `e`
```
JUMP fEnd //go to end of method
```

Startup Code for SaM

- On a real machine
  - OS transfers control to the `main` routine.
  - Control returns to the OS when `main` terminates.
- In SaM, it is convenient to begin execution with a standard startup code sequence that sets up the stack frame for `main` and calls `main`.
  - This allows us to treat `main` like any other method.

```
//Startup code sequence to set up call to main

PUSHIMM 0 //rv slot for main
LINK //save FBR
JSR main //call main
POPFBR
STOP
```