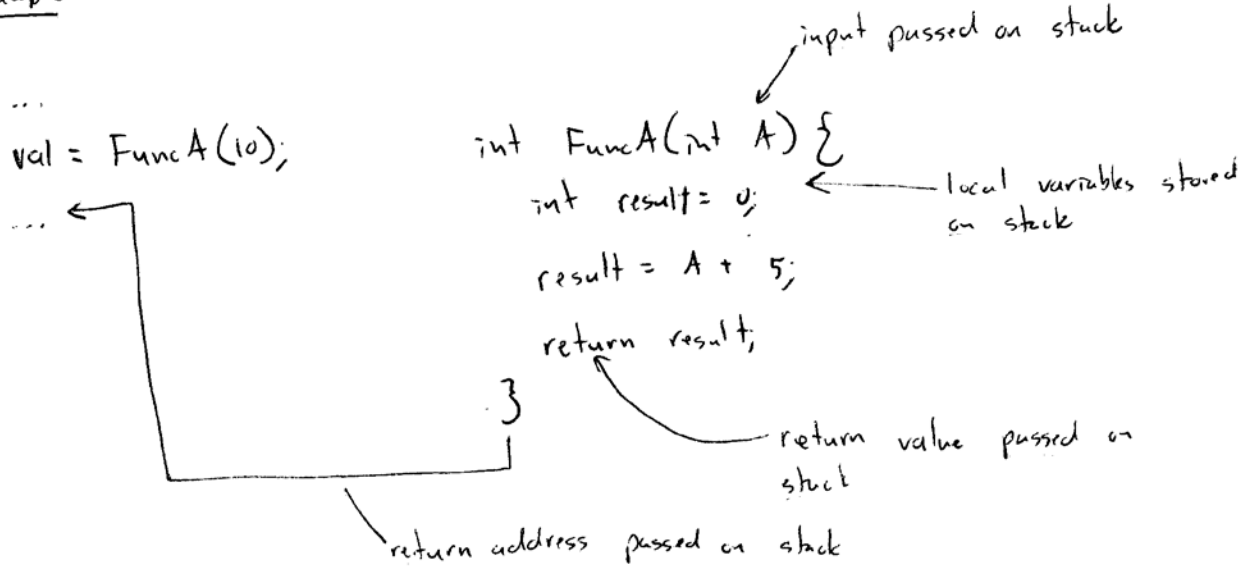
C Stack:

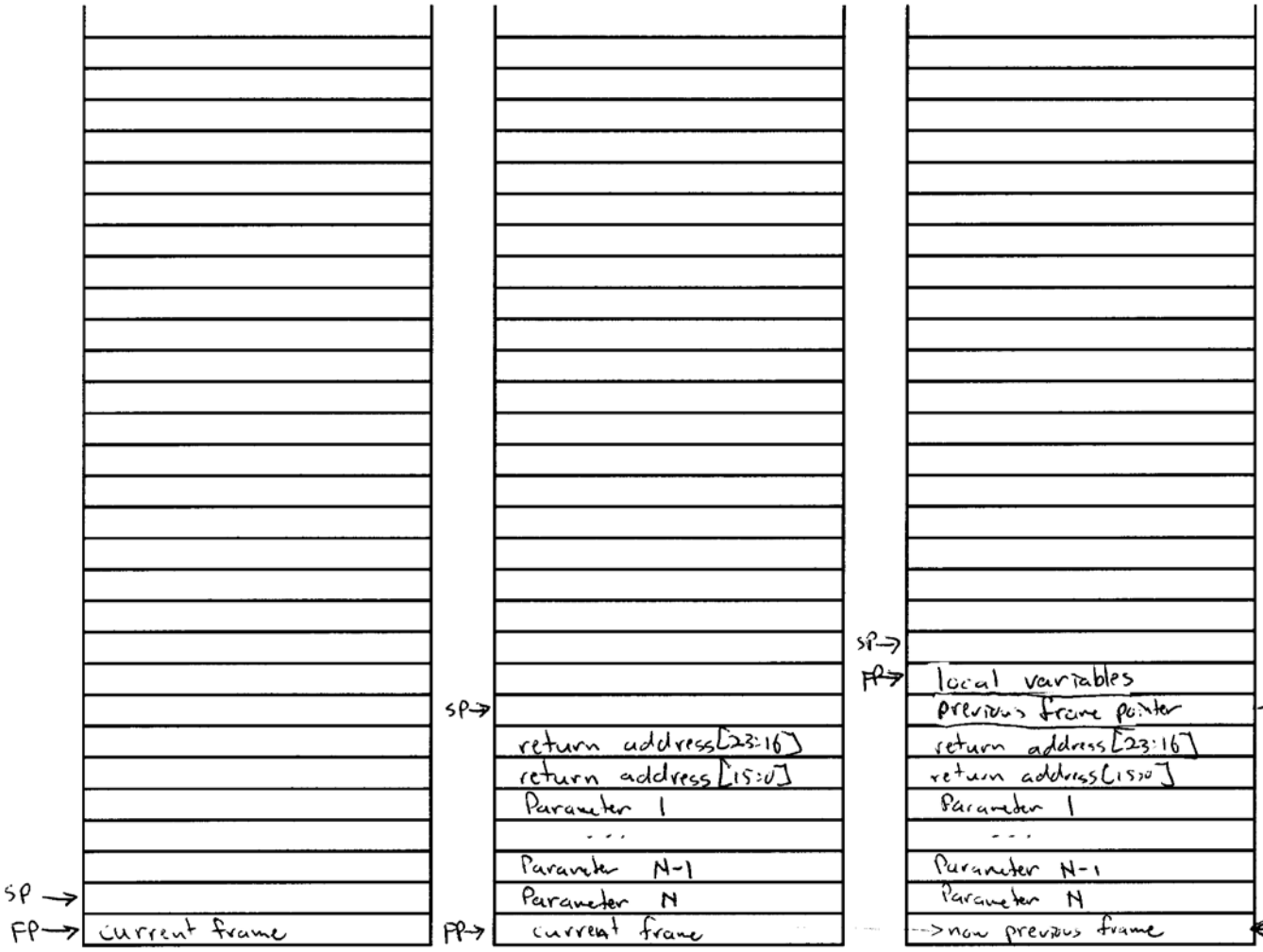
- Used for passing and returning values to/from functions
- Used for returning to correct location after return from function call

Example:



Stack Pointer: Points to top of stack where new information is stored.

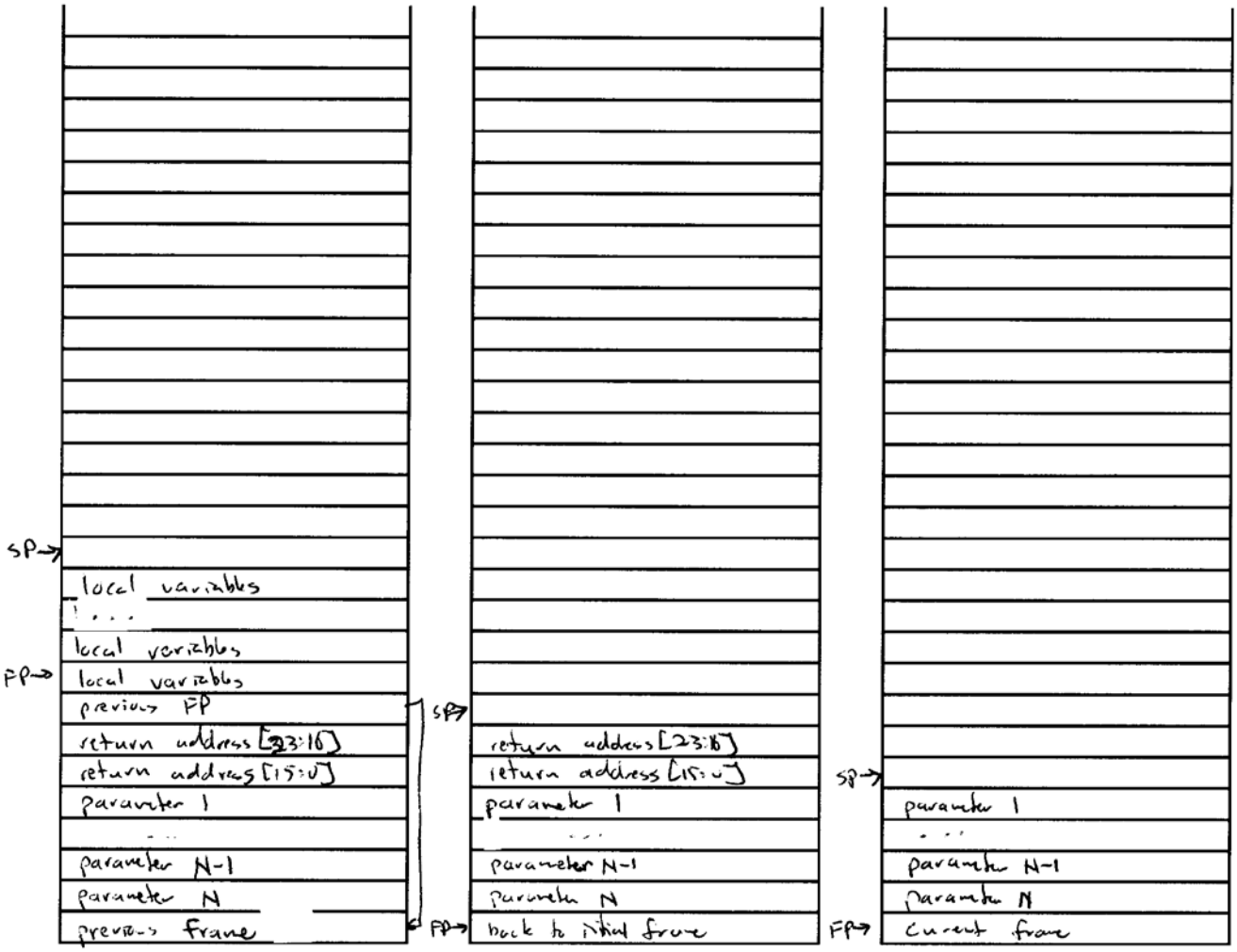
Frame Pointer: Points to current "frame" of stack used by current function executing.



- Initial Configuration:
  - ↳ Before we call the function, we need to pass parameters and save return address on stack

- Ready to call function
  - ↳ After calling function (i.e. a branch instruction), function must setup new frame

- Local variables within function are allocated on stack
- return value can be stored as a parameter

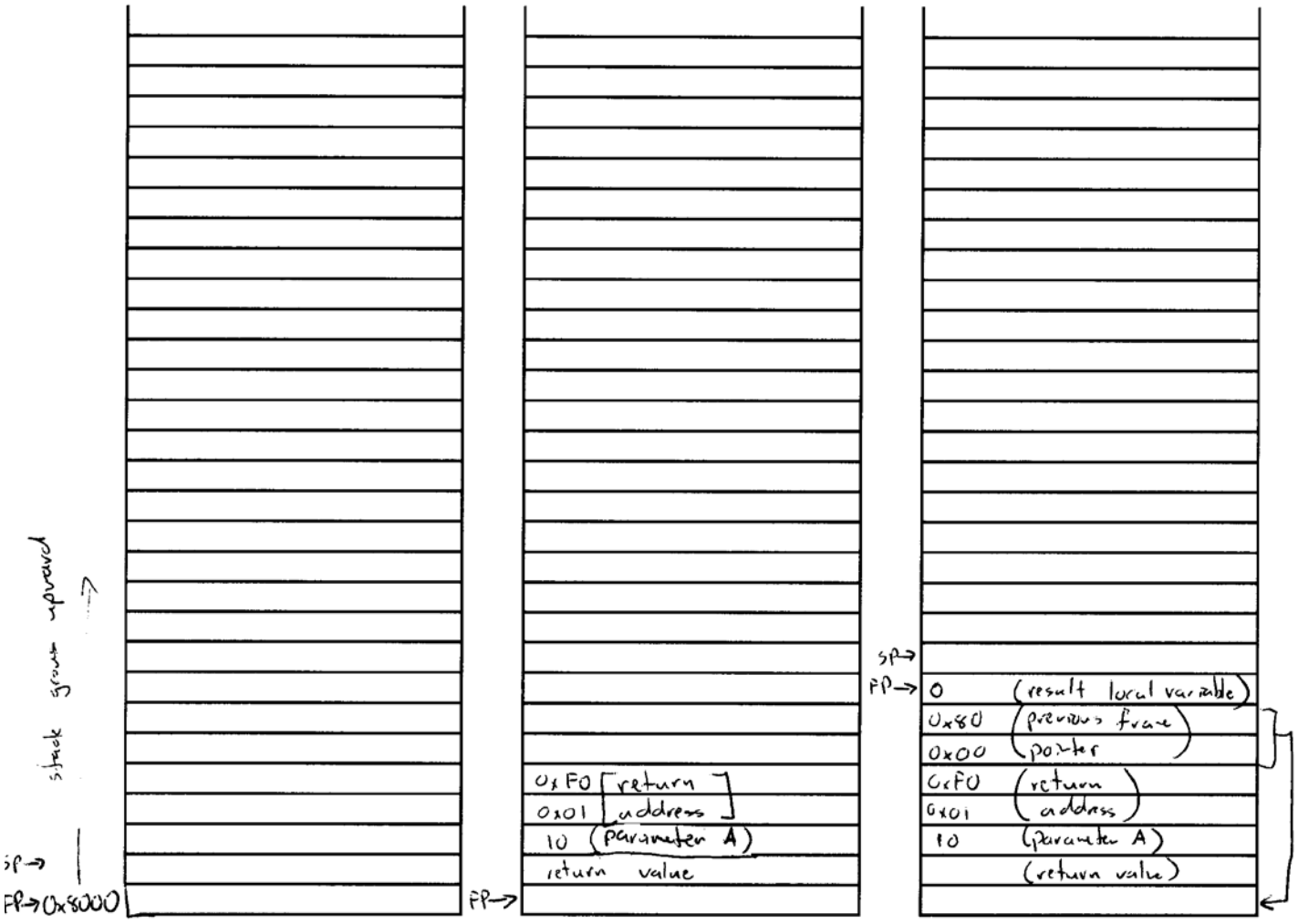


- When function returns:
  1. return value stored in reserved location
  2. pop local variables from stack
  3. return frame pointer to previous frame

- call instruction to jump to return address and update SP

- return value can be found in reserved location
- pop parameters of stack

Data Memory:



Program Memory:

0x EFFF
0x F000 FuncA (10);
0x F001 ← instruction after function call
⋮

```
int FuncA (int A) {
  int result = 0;
  result = A + 5;
  return result;
}
```

① allocate space for return value

② push input parameter (10) on stack

③ store return address (address of instruction after function call)

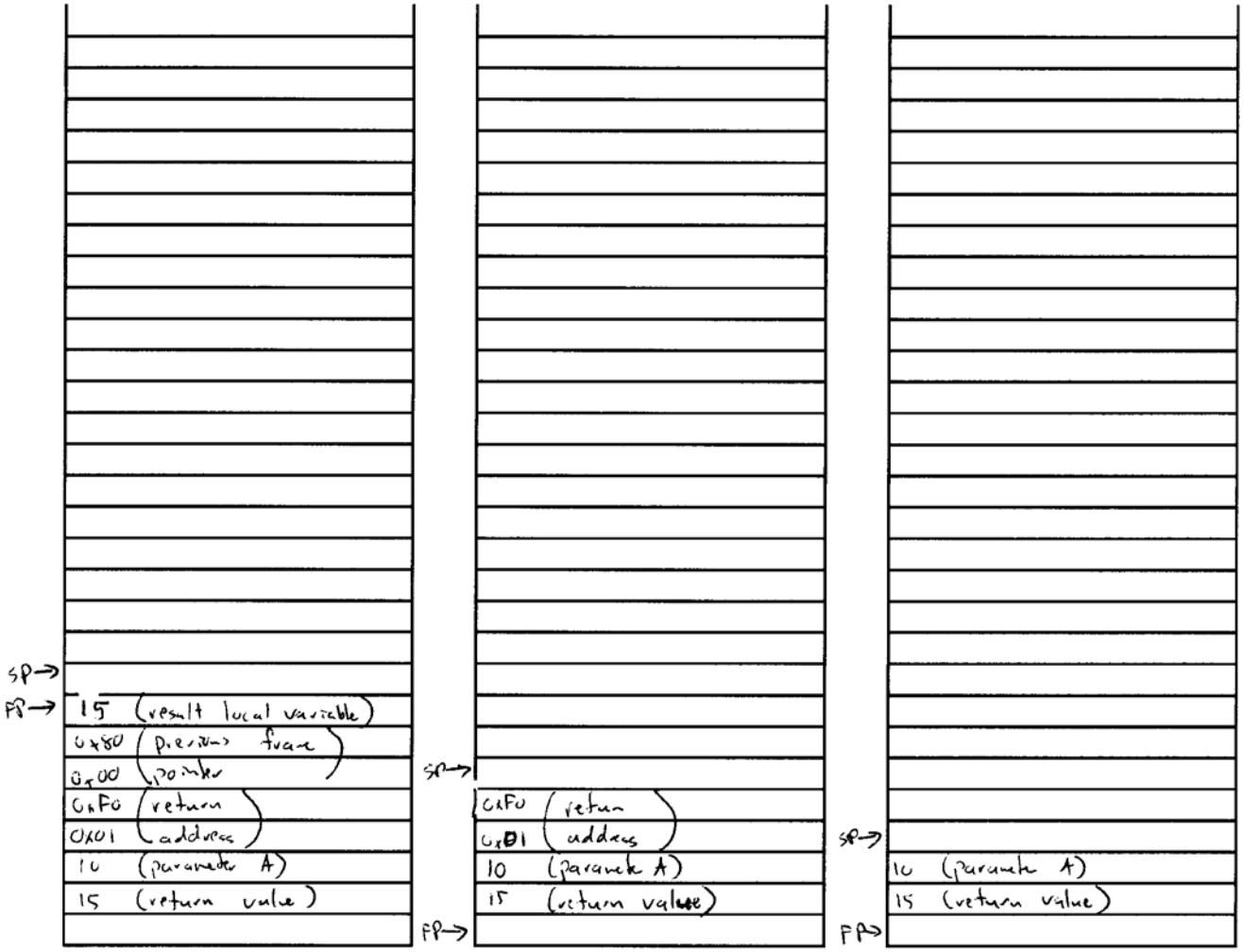
④ jump to FuncA

⑤ store previous frame pointer

⑥ update frame pointer

⑦ push local variable onto stack (e.g. result)

Data Memory:



⑧ update result (e.g. 10+5)

⑨ store result in space reserved for return value

⑩ pop local variable

⑪ set FP to previous value and pop stored frame pointer

⑫ jump to return address

⑬ pop parameter values

⑭ pop return value

Program Memory:

0xEFFF:
0xF000: FuncA(10)
0xF001: ← instruction after function call

```
int FuncA(int A) {
  int result = 0;
  result = A + 5;
  return result;
}
```