T4 (on A3)

Changes to the c6 language (to make it easier to implement)

- Function definition can start with the keyword “func”.
- Inside a function, “return <expr>” returns the value of <expr> and exits
  - As before, it translates to “ret” in nas
  - Please make sure that if a function doesn’t have any “return”, you will insert a “ret” in the end
- I let my “functions” to be put anywhere in the source program
  - I added a “jmp” to jump over it so that it won’t be executed as part of the main program
- If you prefer, you could put the main program inside main() { …}
  - I didn’t do that

Changes in nas

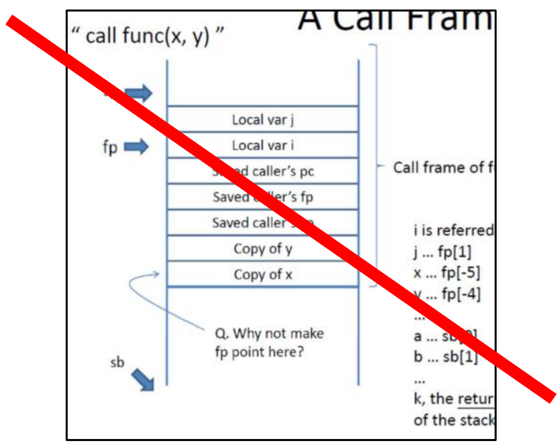
- Added the “var” (for “variables”) instruction
  - Before, to call a function in nas, we need … “call func, 2”, where 2 is the number of arguments to pass to the function
  - Now, we just “call func”, but the first instruction of the function would be “var 2, 1” where 2 is the number of arguments, and 1 is the number of local variables created/used inside the function
  - A total of 3 in this example, for which the compiler needs to reserve stack space
  - This simplifies the implemention, and the new frame design look like this

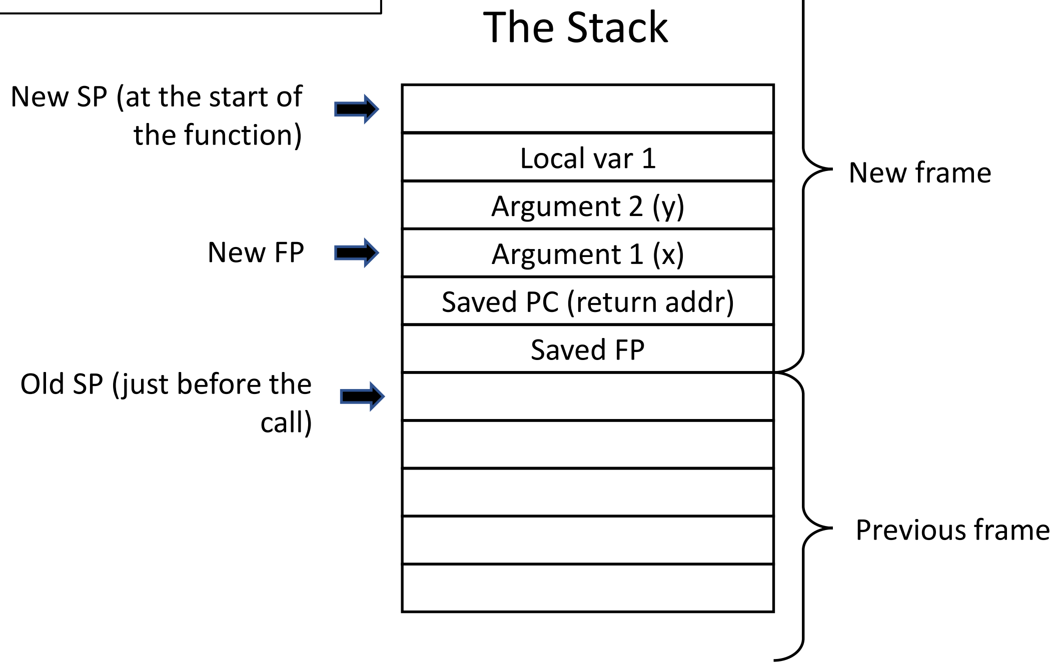
C6

```
func F(a, b) { … }
…
…
puti(F(c, d))   // call F()
…
```

nas2

```
push x
push y
call L501
puti    // prints returned value
…
…
L501: var 2,1    // func begins
…
ret    // func ends
```





- Inside a function in c6, global variables are distinguished by “@”
- The changed nas is called “nas2”

```
a = 123;
b = 456;
func foo(a) {
   a = 100;
   @b = @b + 1;
   b = 789;
}
foo(a);
puti(a);   // 123
puti(b);   // 457
```

```
puts("This is recursive factorial!");
ok = 0;
while (ok == 0) {
  puts_("Please input a number between
0 and 20: ");
  geti(input);
  if (input >= 0 && input <= 20) ok = 1;
}
puti(fact(input));
func fact(n) {
  if (n==0) return 1;
  else return n*@fact(n-1);
```