5.3 Function code generation

EXERCISE #5 ▶ **Function code generation**
Generate the code for the program (use R0 for parameters, R6 for the stack, R7 is reserved for storing the PC):

```
function (x:int) {
    var y:int
    y:=5
    returns (x+y) }
```

```
main(){
    z:=f(12)+1
}
```

Solution. Cf les slides du cours. La génération de code pour les fonctions est standard, on utilise la pile pour passer paramètres et adresse de retour. La spécificité du LC3 est que l'on peut pousser l'adresse de retour dans la fonction appelée parce que le call (JSR) met cette adresse dans R7. J'utilise PUSH comme macro qui fait $STRR6$ puis incrémente de 1 R6. POP est symétrique.

On commence par le main, puis f:

```
main: PUSH(R0,R1,....)    #sauvegardes ...
      AND tmp1 temp1 0
      ADD tmp1 temp1 12   #expression 12
      PUSH tmp1           #pousse l'argument sur la pile
      JSR f #effectue le call et R7 devient l'@ de retour
      AND tmp3 temp3 0
      ADD tmp3 temp3 R0   #récupération du résultat pour la suite
      [flemme pour le reste]

f:     PUSH R7                 #sauve l@ de retour
       ADD R7 R6 0
       ADD R6 R7 xx            #place pour les variables spillées
       LDR tmp1 R7 #-1         #récup de l'argument
       AND tmp2 tmp2 0
       ADD tmp2 tmp2 5
       ADD tmp3 tmp2 tmp1      #corps de la fonction
       ADD R0 tmp3 0           #return value
       ADD R6 R6 -xx           #postlude
       POP R7
       ADD R6 R6 -1
       RET
```

En faisant l'allocation, on n'aura pas de variable spillées, donc xx=0.

□