Stacks
Conventions:
  - Dedicate a register for the Stack Point (SP), R29
  - Builds UP (towards higher addresses) on push
  - SP points to first UNUSED location
Management Macros
  - PUSH(RX): push Reg[x] onto stack
    `ADDC(R29,4,R29)`
    `ST(RX,-4,R29)`
  - POP(RX): pop the value on the top of the stack into Reg[x]
    `LD(R29,-4,RX)`
    `ADDC(R29,-4,R29)`
  - ALLOCATE(k): reserve k words of stack
    `ADDC(R29,4*k,R29)`
  - DEALLOCATE(k): release k words of stack
    `SUBC(R29,4*k,R29)`

Procedure Linkage Contract:
The CALLER will:
  - Push args onto stack in reverse order
  - Branch to callee, putting return address into LP
  - Remove args from stack on return
The CALLEE will:
  - Perform promised computation, leaving result in R0
  - Branch to return address
  - Leave stacked data intact, including stacked args
  - Leave regs (except R0) unchanged

Special Registers
  - R27 = BP Base ptr points into stack to local variables of callee
  - R28 = LP Linkage ptr is return address to caller
  - R29 = SP Stack ptr points to first unused word
(and we already know)
  - R31 = 0 Fixed to zero
Problem 1:

gcd:

```
PUSH (LP)
PUSH (BP)
MOVE (SP, SP)
PUSH (R1)
PUSH (R2)
LD (BP, -12, R0)
LD (BP, -16, R1)
CMPEQ (R0, R1, R2)
BT (R2, L1)
CMPLE (R0, R1, R2)
BT (R2, L2)
PUSH (R1)
SUB (R0, R1, R2)
PUSH (R2)
BR (gcd, LP)
DEALLOCATE (2)
BR (L1)
```

```
L2:
SUB (R1, R0, R2)
PUSH (R2)
PUSH (R0)
BR (gcd, LP)
DEALLOCATE (2)
```

```
L1:
POP (R2)
POP (R1)
MOVE (BP, SP)
POP (BP)
POP (LP)
JMP (LP)
```