Program #1
(see class_cpp.pdf)

H(R1, 5)
Reg: send R1; Memory: write to "a"
Reg: send R1; Display:

Program #2
H(R1, 5)
Reg: send R1; Mem: write to "a"
H(R3, 7)
Reg: send R3; Mem: write to "a"
Reg: send R1; Mem: write to "a"
Reg: send R1; Display

Program #3
H(R1, 5)
Reg: send R1; Mem: write to "a"
H(R3, 17)
Reg: send R3; Mem: write to "b"

If 'negative jump

Reg: send R1; Mem: write to "a"

Reg: send R1; Display
Program #4

H(R1, 0)
Reg 1: Send R1, Mem: write to "4"
H(R3, 0)
Reg 3: Send R3, Mem: write to "i"
H(R2, 4)
H(R0, 1)

U
IF, negative Jump
A
Reg 1: Send R1, Mem: write to "a"
A'
Reg 3: Send R3, Mem: write to "i"
Jump?

Reg 3: Send R1, display
**Program #5**

\( H(R1, 5) \)

Reg: Send R1; Mem: write to "a"

\( H(R3, 100) \)

C

Reg: R1; Mem: write to "b"

Reg: R1; Display

---

Note: 500 does not fit in "b". (Max value is 127), so these most significant bits are lost.

**Program #6**

\( H(R1, 5) \)

Reg: Send R1; Mem: write to "a"

\( H(R3, 100) \)

C

Reg: R0 (and); Mem: write to upper byte of "b"

Reg: Send R1; Mem: write to lower byte of "b"

Reg: Send R0; Display

Reg: Send R1; Display

---

Note: Variable "b" is now 2 bytes can "upper" and a "lower byte"

the value (assuming a positive integer) is:

\[ U \times 256 + L \]

where \( U \) & \( L \) are bytes
Program #7

Assume a new operator!

\[ C' \quad R_0 \times R_3 \rightarrow [R_3, R_5] \quad \text{Multiply} \]

- H(R_{1,0})
  - Reg: Send R1; Mem: Write to \( q' \) \[ R_0 := 2 \]
  - Reg: Send R1; Mem: Write to \( q' \)

- H(R_{3,1})
  - Reg: Send R3; Mem: Write to \( q_3' \)

- H(R_{0,2})

- H(R_{2,0})
  - U
    - J if Zero
      - Jump

- Reg: Send R3; Mem: Write to \( q_3' \)
- Reg: Send R1; Mem: Display

\[ C' \]

- J(R_{3, R_5})
  - Copy result to R3

No more code.