LC-3

More LC-3 Programming

Porter 256 does not exist, it is in 246

New session Monday 4PM ARC 116
Control Instructions

Used to alter the sequence of instructions. This allows us to move to a particular instruction.

Conditional Branch

- branch is *taken* if a specified condition is true
- else, the branch is *not taken*
  - next sequential instruction is executed

\[
\text{if } a < b
\]
Unconditional Branch (or Jump)
- always changes instruction

TRAP
- changes to an OS “service routine”
- routine will return control to the next instruction (after TRAP) when finished
Condition Codes

LC-3 has three condition code bits:

- \( N \) -- negative
- \( Z \) -- zero
- \( P \) -- positive (greater than zero)

Set by any instruction that writes a value to a register (ADD, AND, NOT, LD, LDR, LDI, LEA)

Exactly one will be set at all times — Based on the last instruction that altered a register
AND R0, R0, 0

ADD R1, R0, 0
Branch Instruction

- Branch specifies one or more condition codes.
- If the set bit is specified, the branch is taken.
  - PC-relative addressing is used
  - target address is made by adding signed offset (IR[8:0]) to current PC.
If the branch is not taken, the next sequential instruction is executed.

There are hardware limits on how far you can branch.
BR (Branch)

- It does no computation, only looks at condition codes
- If condition code is set, go to LABEL, can combine codes
  - BRz FOO
  - BRn Bar
  - BRp FOO
  - BRzp Queen

ADD R0, R1, R2

BRz FOO ; if zero we go to label FOO
BR (unconditionally)

- Degenerate case, always goes to LABEL
  - BRnzp

BRnzp FOO
BRnzp FOO ; always go to label FOO

BR

BRnzp 

↑
Example: Using a Branch

Compute sum of 12 integers
Numbers start at label NUMS. Program starts at location x3000.

```
R1 ← NUMS
R3 ← 0
R2 ← 12

R2=0?  NO  YES

R4 ← M[R1]
R3 ← R3+R4
R1 ← R1+1
R2 ← R2-1
```
1, 3, 5, 7, 9

R4 \leftrightarrow 4 \leftrightarrow 3 \leftrightarrow S

12 \quad N2P

\frac{3}{2} \quad \downarrow \quad 12

12 - 12 = 0

0
## Example: Using a Branch

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>LEA R1,NUMS</code></td>
<td><code>R1 ← NUMS</code></td>
</tr>
<tr>
<td><code>AND R3,R3,0</code></td>
<td><code>R3 ← 0</code></td>
</tr>
<tr>
<td><code>AND R2,R2,0</code></td>
<td><code>R2 ← 0</code></td>
</tr>
<tr>
<td><code>ADD R2,R2,12</code></td>
<td><code>R2 ← 12</code></td>
</tr>
<tr>
<td><code>START</code></td>
<td><code>If Z, goto END</code></td>
</tr>
<tr>
<td><code>BRnzp START</code></td>
<td><code>Goto START</code></td>
</tr>
<tr>
<td><code>END</code></td>
<td><code># done adding</code></td>
</tr>
<tr>
<td><code>LDR R4,R1,0</code></td>
<td><code>Load next value to R4</code></td>
</tr>
<tr>
<td><code>ADD R3,R4,R3</code></td>
<td><code>R3 ← R4 + R3</code></td>
</tr>
<tr>
<td><code>ADD R1,R1,1</code></td>
<td><code>Increment R1 (pointer)</code></td>
</tr>
<tr>
<td><code>ADD R2,R2,-1</code></td>
<td><code>Decrement R2 (counter)</code></td>
</tr>
</tbody>
</table>

| `N2P`               |                               |
H \times 65 \quad 78
Instructions

JMP

Jump is an unconditional branch -- *always* taken.
- Target is contents of a register, not a label.

\[
\text{JMP R1}
\]

\[
\text{LEA R1, FOO}
\]

\[
\text{JMP R1}
\]
LC-3 Subtraction

\begin{align*}
\text{inverse} \\
R_0 &= \overline{R_0} \\
R_0 &= R_0 + 1 \\
R_2 &= R_0 + R_1
\end{align*}

\text{NOT } R_0, R_0 \\
\text{ADD } R_0, R_0, 1
Print Single Digit Number

RO = 7

ASCII offset = 48
LD R1, AOFF
ADD RO, RO, R1
OUT

AOFF .FILL 48 '0'
LC-3 Multiply

R1 ← 5
R3 ← 0
R2 ← 16

if R2 = 0?

R3 ← R3 + R1
R2 ← R2 - 1

R2 ← abs(16)
LC-3 Integer Division

\[
\frac{20}{3} \quad 20 - 3 \ldots -3
\]

32-bit + 2 instructions
\[\div 3\]
\[ \frac{N}{36} \]

\[ N \times \text{millions} \]

8-bit

16-bit + 5

\[ \text{division} \]
Masks

\[
\begin{array}{cccccccc}
1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
BRz & F00
\end{array}
\]