Control Instructions

Altering sequence of instructions
- PC is always incremented in FETCH stage
  - By default we always move to next instruction
  - An instruction can alter the flow of the program
    - Then PC + 1 value will be overridden

1. Conditional Branch
- High-level construct: conditional statement
- Branch *taken* if a specified condition is true
  - New PC computed relative to current PC
- Otherwise, branch *not taken*
  - PC is unchanged (i.e., points to next sequential instruction)

2. Unconditional Branch (or Jump)
- High-level construct: return, break
- Always changes the PC
- Target address computed PC-relative or Base+Offset

Condition Codes

LC-3 has three 1-bit condition code registers
- N -- negative
- Z -- zero
- P -- positive (greater than zero)

Set/cleared by instructions that store value to register
- Operate and Load instructions
- e.g., ADD, AND etc.

Exactly one will be set at all times
- Based on the last instruction that altered a register

PC Relative Addressing Mode

Want to specify address directly in the instruction
- But an address is 16 bits, and so is an instruction!
  - After subtracting 4 bits for opcode we are left with 12 bits
    - We still need to encode NZP so we are actually left with 9 bits

Bits [8:0] of IR is use to encode information
- Can be positive or negative value
  - Therefore we go +/- from the current PC position
- Hence PC-relative
  - What is the max range of addresses we span from PC?

Forming an address
- Sign extend 9 bits
- Add it to the PC of next instruction to form address
  - Because once the instruction is fetched from memory at location indicated by PC, PC is incremented by 1 (default)
BR Instruction Encoding Example

Give the encoding of a Branch instruction located at x3009 that always changes PC to x3005

LC3 Machine Code for Multiply Operation

<table>
<thead>
<tr>
<th>Address in Hex</th>
<th>LC3 Machine Code</th>
<th>LC3 Machine Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0101 011 011 1 0000</td>
<td>R3&lt;- R3 AND 0 (C = 0)</td>
</tr>
<tr>
<td>0001</td>
<td>0001 010 010 1 0000</td>
<td>R2&lt;- R2 + 0</td>
</tr>
<tr>
<td>0002 (LOOP)</td>
<td>0000 010 00000011</td>
<td>If R2 == 0 then GO TO DONE</td>
</tr>
<tr>
<td>0003</td>
<td>0001 011 011 000 001</td>
<td>R3 &lt;- R3 + R1</td>
</tr>
<tr>
<td>0004</td>
<td>0001 010 010 1 11111</td>
<td>R2 &lt;- R2 – 1</td>
</tr>
<tr>
<td>0005</td>
<td>0000 111 11111100</td>
<td>GO TO LOOP</td>
</tr>
<tr>
<td>0008 (DONE)</td>
<td>1111111111111111</td>
<td>Location of Done (statement after loop)</td>
</tr>
<tr>
<td>00101</td>
<td>AND</td>
<td></td>
</tr>
<tr>
<td>00001</td>
<td>ADD</td>
<td></td>
</tr>
<tr>
<td>00000</td>
<td>BR</td>
<td></td>
</tr>
</tbody>
</table>