Multiple-Cycle Design

- The Multiple-Cycle Implementation demonstrates the use of a single memory for:
  - Data
  - Instruction

- This design is also used to show the implementation of more complex instructions
The following address sources are used to fetch:

- Instructions -> PC Program Counter Register (16bit)
- Data -> Bus A (16bit)
- MUX M selects between the two address sources through the MM control signal
Instructions are executed over multiple clock cycles.

This requires an additional register

- R8 for temporary storage

This register should be selected through an additional bit control signals:

- TD, TA, TB

These control signals are to the left of:

- SA, SB, DR (from IR register)
Instructions must be held in a register during the execution of multiple micro-ops.

The IR is only loaded if an instruction is fetched from memory M.

The IR has a load enable control signal IL.

This signal is part of the control word.
The PC only increments if an instruction is fetched from memory M.

The control word has two bits that determine the PC modifications:

- PI - increment enable signal
  - PC ← PC + 1
- PL – PC load signal
  - PC ← PC + se AD
Next Address Logic

- The **CAR** Control Address Register selects the control word in the 256x 28 control memory.

- The next logic (**MUX S**) determines whether **CAR** is incremented on loaded.
  - Controlled with **MS**

- The source of the loaded address is determined by **MUX C**
  - Selected by **MC**
The sources for the multiplexer can be:
- Contents of the 8 bit **NA** Next Address field
- 7 bit from the opcode field in the **IR**
- An opcode loaded into the **CAR** points to:
  - Microprogram in Control Memory
  - This program implements the instruction through the execution of micro operations
- **MUX S** determines whether the **CAR** is:
  - Incremented
  - Loaded
# Sequencer Control Fields

<table>
<thead>
<tr>
<th>Action</th>
<th>Symbolic Notation Code</th>
<th>Select</th>
<th>Symbolic Notation</th>
<th>Action</th>
<th>Symbolic Notation</th>
<th>Action</th>
<th>Symbolic Notation</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increment CAR</td>
<td>CNT</td>
<td>000</td>
<td>NA</td>
<td>NXA</td>
<td>No load</td>
<td>NLI</td>
<td>No load</td>
<td>NLP  0</td>
</tr>
<tr>
<td>Load CAR</td>
<td>NXT</td>
<td>001</td>
<td>Opcode</td>
<td>OPC</td>
<td>Load instr.</td>
<td>LDI</td>
<td>Increment PC</td>
<td>INP  1</td>
</tr>
<tr>
<td>If $C = 1$, load CAR; else increment CAR</td>
<td>BC</td>
<td>010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>If $V = 1$, load CAR; else increment CAR</td>
<td>BV</td>
<td>011</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>If $Z = 1$, load CAR; else increment CAR</td>
<td>BZ</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>If $N = 1$, load CAR; else increment CAR</td>
<td>BN</td>
<td>101</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>If $C = 0$, load CAR; else increment CAR</td>
<td>BNC</td>
<td>110</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>If $Z = 0$, load CAR; else increment CAR</td>
<td>BNZ</td>
<td>111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>