ARM Addressing Modes
Addressing Modes

• Addressing refers to means to specify location of operands for instructions
  - Types of addressing are called addressing modes

• Classification
  - Immediate
  - Direct
  - Register Direct
  - Indirect
  - Register Indirect
  - Displacement (Indexed)
  - Stack
Immediate Addressing

- Operand is part of instruction
- Operand = address field
- e.g. ADD 5
  - Add 5 to contents of accumulator
  - 5 is operand
  - No memory reference to fetch data
  - Fast
  - Limited range

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Operand</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Direct Addressing

• **Absolute addressing**
  • Address field contains address of operand

• **Effective address:** the actual address of data or an instruction in main storage
  • Effective address (EA) = address field (A)
  • e.g. ADD A
    - Add contents of cell A to accumulator
    - Look in memory at address A for operand

• Single memory reference to access data

• No additional calculations to work out effective address

• Limited address space
Direct Addressing Diagram

Opcode | Address A

Memory

Operand
Register Direct Addressing (1)

- Operand is held in register named in address filed
- EA = R
- Limited number of registers
- Very small address field needed
  - Shorter instructions
  - Faster instruction fetch
Register Direct Addressing (2)

• No memory access
• Very fast execution
• Very limited address space
• Multiple registers helps performance
  - Requires good assembly programming or compiler writing
  - C programming
    • register int a;
Register Addressing Diagram

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Register R</th>
</tr>
</thead>
</table>

Registers

Operand
Indirect Addressing (1)

- Memory cell pointed to by address field contains the address of (pointer to) the operand
- \( EA = (A) \)
  - Look in A, find address \((A)\) and look there for operand
- e.g. \( ADD \ (A) \)
  - Add contents of cell pointed to by contents of A to accumulator
Indirect Addressing (2)

- Large address space
- May be nested, multilevel, cascaded
- e.g. EA = ((((A))))
- Draw the diagram yourself
- Multiple memory accesses to find operand
- Hence slower
Indirect Addressing Diagram

| Opcode | Address A |

Memory

Pointer to Operand

Operand
Register Indirect Addressing

• $EA = (R)$
• Operand is in memory cell pointed to by contents of register $R$
• Large address space
• One fewer memory access than indirect addressing
Register Indirect Addressing Diagram

Opcode \( R \)

Registers

Pointer to Operand

Memory

Operand
Displacement Addressing

- \( EA = (R) + o \)
- **Address field hold two values**
  - \( R = \) register that holds the base address
  - \( o = \) offset
  - or vice versa
Relative Addressing

• A version of displacement addressing
• \( R = \text{Program counter, } PC \)
• \( EA = (PC) + \text{offset} \)
• i.e. get operand from offset and from current location pointed to by PC
### Stack Addressing

- The operand is (implicitly) on the top of the stack.

<table>
<thead>
<tr>
<th>Name</th>
<th>Stack</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-increment load</td>
<td>LDMED</td>
<td>LDMIB</td>
</tr>
<tr>
<td>Post-increment load</td>
<td>LDMFD</td>
<td>LDMIA</td>
</tr>
<tr>
<td>Pre-decrement load</td>
<td>LDMEA</td>
<td>LDMDB</td>
</tr>
<tr>
<td>Post-decrement load</td>
<td>LDMFA</td>
<td>LDMDA</td>
</tr>
<tr>
<td>Pre-increment store</td>
<td>STMFA</td>
<td>STMIB</td>
</tr>
<tr>
<td>Post-decrement store</td>
<td>STMED</td>
<td>STMDA</td>
</tr>
<tr>
<td>Post-increment store</td>
<td>STMEA</td>
<td>STMIA</td>
</tr>
<tr>
<td>Pre-decrement store</td>
<td>STMFD</td>
<td>STMDB</td>
</tr>
</tbody>
</table>