5.3 Instruction types

Instructions fall into several broad categories that you should be familiar with:

- Data movement.
- Arithmetic.
- Boolean.
- Bit manipulation.
- I/O.
- Control transfer.
- Special purpose.

Can you think of some examples of each of these?

Data Transfer

- Specify
  - Source
  - Destination
  - Amount of data
- May be different instructions for different movements
  - e.g. IBM 370
- Or one instruction and different addresses
  - e.g. VAX

Arithmetic

- Add, Subtract, Multiply, Divide
- Signed Integer
- Floating point?
- May include
  - Increment (a++)
  - Decrement (a–)
  - Negate (-a)

Logical

- Bitwise operations
- AND, OR, NOT
Conversion
- E.g. Binary to Decimal

Input/Output
- May be specific instructions
- May be done using data movement instructions (memory map ped)
- May be done by a separate controller (DMA)

Systems Control
- Privileged instructions
- CPU needs to be in specific state
  - Ring 0 on 80386+
  - Kernel mode
- For operating systems use

Transfer of Control
- Branch
  - e.g. branch to x if result is zero
- Skip
  - e.g. increment and skip if zero
  - ISZ Register1
  - Branch xxxx
  - ADD A
- Subroutine call
  - e.g. interrupt call

5.4 Addressing
- Addressing modes specify where an operand is located.
- They can specify a constant, a register, or a memory location.
- The actual location of an operand is its effective address.
- Certain addressing modes allow us to determine the address of an operand dynamically.

5.4 Addressing
- Immediate addressing is where the data is part of the instruction.
- Direct addressing is where the address of the data is given in the instruction.
- Register addressing is where the data is located in a register.
- Indirect addressing gives the address of the address of the data in the instruction.
- Register indirect addressing uses a register to store the address of the address of the data.
5.4 Addressing

- **Indexed addressing** uses a register (implicitly or explicitly) as an offset, which is added to the address in the operand to determine the effective address of the data.
- **Based addressing** is similar except that a base register is used instead of an index register.

The difference between these two is that an index register holds an offset relative to the address given in the instruction, a base register holds a base address where the address field represents a displacement from this base.
5.4 Addressing

- In *stack addressing* the operand is assumed to be on top of the stack.
- There are many variations to these addressing modes including:
  - Indirect indexed.
  - Base/offset.
  - Self-relative
  - Auto increment - decrement.
- We won’t cover these in detail.

Let’s look at an example of the principal addressing modes.

For the instruction shown, what value is loaded into the accumulator for each addressing mode?

<table>
<thead>
<tr>
<th>Mode</th>
<th>Value Loaded into AC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Immediate</td>
<td></td>
</tr>
<tr>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>Indirect</td>
<td></td>
</tr>
<tr>
<td>Indexed</td>
<td></td>
</tr>
</tbody>
</table>

These are the values loaded into the accumulator for each addressing mode.

<table>
<thead>
<tr>
<th>Memory</th>
<th></th>
<th>LOAD 800</th>
</tr>
</thead>
<tbody>
<tr>
<td>800</td>
<td>900</td>
<td></td>
</tr>
<tr>
<td>900</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>500</td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td>600</td>
<td></td>
</tr>
<tr>
<td>1600</td>
<td>700</td>
<td></td>
</tr>
</tbody>
</table>