Chapter 11
Instruction Sets:
Addressing Modes
and Formats
Addressing Modes

- Immediate
- Direct
- Indirect
- Register
- 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
**Immediate Addressing Diagram**

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

• Address field contains address of operand
• 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

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Address A</th>
</tr>
</thead>
</table>

Instruction

Memory

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
- \(2^n\) where \(n = \text{word length}\)
- May be nested, multilevel, cascaded
  - e.g. \(EA = (((A)))\)
    - Draw the diagram yourself
- Multiple memory accesses to find operand
- Hence slower
Indirect Addressing Diagram

Instruction

| Opcode | Address A |

Memory

Pointer to operand

Operand
Register Addressing (1)

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

- No memory access
- Very fast execution
- Very limited address space
- Multiple registers helps performance
  - Requires good assembly programming or compiler writing
  - N.B. C programming
    - register int a;
- c.f. Direct addressing
Register Addressing Diagram

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

Instruction

Registers

Operand
Register Indirect Addressing

- C.f. indirect addressing
- EA = (R)
- Operand is in memory cell pointed to by contents of register R
- Large address space ($2^n$)
- One fewer memory access than indirect addressing
Register Indirect Addressing Diagram

Instruction

| Opcode | Register Address R |

Registers

Pointer to Operand

Memory

Operand
Displacement Addressing

- \( EA = A + (R) \)
- Address field hold two values
  - \( A = \) base value
  - \( R = \) register that holds displacement
  - or vice versa
Displacement Addressing Diagram

Instruction

Opcode | Register R | Address A

Registers

Pointer to Operand

+ 

Memory

Operand
Relative Addressing

- A version of displacement addressing
- \( R = \) Program counter, PC
- \( EA = A + (PC) \)
- i.e. get operand from A cells from current location pointed to by PC
- c.f locality of reference & cache usage
Base-Register Addressing

- A holds displacement
- R holds pointer to base address
- R may be explicit or implicit
- e.g. segment registers in 80x86
Indexed Addressing

- $A = \text{base}$
- $R = \text{displacement}$
- $EA = A + R$
- Good for accessing arrays
  - $EA = A + R$
  - $R++$
Combinations
  • Postindex
  • $EA = (A) + (R)$

  • Preindex
  • $EA = (A+(R))$

  • (Draw the diagrams)
Stack Addressing

- Operand is (implicitly) on top of stack
- e.g.
  - ADD Pop top two items from stack and add
Pentium Addressing Modes

• Virtual or effective address is offset into segment
  — Starting address plus offset gives linear address
  — This goes through page translation if paging enabled

• 12 addressing modes available
  — Immediate
  — Register operand
  — Displacement
  — Base
  — Base with displacement
  — Scaled index with displacement
  — Base with index and displacement
  — Base scaled index with displacement
  — Relative
PowerPC Addressing Modes

• Load/store architecture
  — Indirect
    - Instruction includes 16 bit displacement to be added to base register (may be GP register)
    - Can replace base register content with new address
  — Indirect indexed
    - Instruction references base register and index register (both may be GP)
    - EA is sum of contents

• Branch address
  — Absolute
  — Relative
  — Indirect

• Arithmetic
  — Operands in registers or part of instruction
  — Floating point is register only
PowerPC Memory Operand Addressing Modes

(a) Indirect Addressing

(b) Indirect Indexed Addressing
Instruction Formats

- Layout of bits in an instruction
- Includes opcode
- Includes (implicit or explicit) operand(s)
- Usually more than one instruction format in an instruction set
Instruction Length

• Affected by and affects:
  — Memory size
  — Memory organization
  — Bus structure
  — CPU complexity
  — CPU speed

• Trade off between powerful instruction repertoire and saving space
Allocation of Bits

• Number of addressing modes
• Number of operands
• Register versus memory
• Number of register sets
• Address range
• Address granularity
PDP-8 Instruction Format

<table>
<thead>
<tr>
<th>Memory Reference Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input/Output Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Register Reference Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 1 Microinstructions</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

| Group 2 Microinstructions       |
| 1 | 1 | 1 | 1 | CLA | SMA | SZA | SNL | RSS | OSR | HLT | 0   |
| 0 | 1 | 2 | 3 | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  |

| Group 3 Microinstructions       |
| 1 | 1 | 1 | 1 | CLA | MQA | 0   | MQL | 0   | 0   | 0   | 1   |
| 0 | 1 | 2 | 3 | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  |

**Definitions**
- **D/L**: Direct/Indirect address
- **Z/C**: Page 0 or Current page
- **CLA**: Clear Accumulator
- **CLL**: Clear Link
- **CMA**: Complement Accumulator
- **CML**: Complement Link
- **RAR**: Rotate Accumulator Right
- **RAL**: Rotate Accumulator Left
- **BSW**: Byte Swap
- **LAC**: Increment Accumulator
- **SMA**: Skip on Minus Accumulator
- **SZA**: Skip on Zero Accumulator
- **SNL**: Skip on Nonzero Link
- **RSS**: Reverse Skip Sense
- **OSR**: Or with Switch Register
- **HLT**: Halt
- **MQA**: Multiplier Quotient into Accumulator
- **MQL**: Multiplier Quotient Load
PDP-10 Instruction Format

I = indirect bit
## PDP-11 Instruction Format

<table>
<thead>
<tr>
<th></th>
<th>Opcode</th>
<th>Source</th>
<th>Destination</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Numbers below fields indicate bit length

Source and Destination each contain a 3-bit addressing mode field and a 3-bit register number

FP indicates one of four floating-point registers

R indicates one of the general-purpose registers

CC is the condition code field
# VAX Instruction Examples

<table>
<thead>
<tr>
<th>Hexadecimal Format</th>
<th>Explanation</th>
<th>Assembler Notation and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 5</td>
<td>Opcode for RSB</td>
<td>RSB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Return from subroutine</td>
</tr>
<tr>
<td>D 4 5 9</td>
<td>Opcode for CLRL</td>
<td>CLRL R9</td>
</tr>
<tr>
<td></td>
<td>Register R9</td>
<td>Clear register R9</td>
</tr>
<tr>
<td>B 0 4 6 0 1</td>
<td>Opcode for MOVW</td>
<td>MOVW 356(R4), 25(R11)</td>
</tr>
<tr>
<td></td>
<td>Word displacement mode,</td>
<td>Move a word from address</td>
</tr>
<tr>
<td></td>
<td>Register R4</td>
<td>that is 356 plus contents</td>
</tr>
<tr>
<td></td>
<td>356 in hexadecimal</td>
<td>of R4 to address that is</td>
</tr>
<tr>
<td></td>
<td>Byte displacement mode,</td>
<td>25 plus contents of R11</td>
</tr>
<tr>
<td></td>
<td>Register R11</td>
<td></td>
</tr>
<tr>
<td></td>
<td>25 in hexadecimal</td>
<td></td>
</tr>
<tr>
<td>C 1 0 5 5 4 2</td>
<td>Opcode for ADDL3</td>
<td>ADDL3 #5, R0, #A[R2]</td>
</tr>
<tr>
<td></td>
<td>Short literal 5</td>
<td>Add 5 to a 32-bit integer in</td>
</tr>
<tr>
<td></td>
<td>Register mode R0</td>
<td>R0 and store the result in</td>
</tr>
<tr>
<td></td>
<td>Index prefix R2</td>
<td>location whose address is</td>
</tr>
<tr>
<td></td>
<td>Indirect word relative,</td>
<td>sum of A and 4 times the</td>
</tr>
<tr>
<td></td>
<td>(displacement from PC)</td>
<td>contents of R2</td>
</tr>
<tr>
<td></td>
<td>Amount of displacement from</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PC relative to location A</td>
<td></td>
</tr>
</tbody>
</table>
Pentium Instruction Format

Instruction prefix | Segment override | Operand size override | Address size override

0 or 1 | 0 or 1 | 0 or 1 | 0 or 1 bytes

Instruction prefixes | Opcode | ModR/M | SIB | Displacement | Immediate

0, 1, 2, 3, or 4 bytes | 1 or 2 | 0 or 1 | 0 or 1 | 0, 1, 2, or 4 | 0, 1, 2, or 4

Mod | Reg/Opcode | R/M

7 6 5 4 3 2 1 0

Scale | Index | Base

7 6 5 4 3 2 1 0
### PowerPC Instruction Formats (1)

<table>
<thead>
<tr>
<th>Branch</th>
<th>Long Immediate</th>
<th>AL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Br Conditional</td>
<td>Options</td>
<td>CR Bit</td>
</tr>
<tr>
<td>Br Conditional</td>
<td>Options</td>
<td>CR Bit</td>
</tr>
</tbody>
</table>

(a) Branch instructions

<table>
<thead>
<tr>
<th>CR</th>
<th>Dest Bit</th>
<th>Source Bit</th>
<th>Source Bit</th>
<th>Add, OR, XOR, etc.</th>
<th>/</th>
</tr>
</thead>
</table>

(b) Condition register logical instructions

<table>
<thead>
<tr>
<th>Ld/St Indirect</th>
<th>Dest Register</th>
<th>Base Register</th>
<th>Displacement</th>
<th>Displacement</th>
<th>Size, Sign, Update</th>
<th>/</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ld/St Indirect</td>
<td>Dest Register</td>
<td>Base Register</td>
<td>Index Register</td>
<td>Size, Sign, Update</td>
<td>/</td>
<td></td>
</tr>
<tr>
<td>Ld/St Indirect</td>
<td>Dest Register</td>
<td>Base Register</td>
<td></td>
<td></td>
<td>Displacement</td>
<td>XO</td>
</tr>
</tbody>
</table>

(c) Load/store instructions
### PowerPC Instruction Formats (2)

#### (c) Load/store instructions

<table>
<thead>
<tr>
<th>Type</th>
<th>Dest Register</th>
<th>Src Register</th>
<th>Src Register</th>
<th>O</th>
<th>Add, Sub, etc.</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD/ST Indirect</td>
<td>Dest Register</td>
<td>Base Register</td>
<td>Index Register</td>
<td>Size, Sign, Update</td>
<td>/</td>
<td></td>
</tr>
<tr>
<td>Load</td>
<td>Dest Register</td>
<td>Base Register</td>
<td>Displacement</td>
<td></td>
<td></td>
<td>XO*</td>
</tr>
</tbody>
</table>

#### (d) Integer arithmetic, logical, and shift/rotate instructions

<table>
<thead>
<tr>
<th>Type</th>
<th>Dest Register</th>
<th>Src Register</th>
<th>Src Register</th>
<th>Src Register</th>
<th>Shift Type or Mask</th>
<th>S</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add, Sub, etc.</td>
<td>Dest Register</td>
<td>Src Register</td>
<td></td>
<td></td>
<td>Signed Immediate Value</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Logical</td>
<td>Src Register</td>
<td>Dest Register</td>
<td>Src Register</td>
<td>ADD, OR, XOR, etc.</td>
<td>Unsigned Immediate Value</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>AND, OR, etc.</td>
<td>Src Register</td>
<td>Dest Register</td>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Rotate</td>
<td>Src Register</td>
<td>Dest Register</td>
<td>Shift Amt</td>
<td>Mask Begin</td>
<td></td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Rotate or Shift</td>
<td>Src Register</td>
<td>Dest Register</td>
<td>Src Register</td>
<td>Shift Type or Mask</td>
<td></td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Rotate</td>
<td>Src Register</td>
<td>Dest Register</td>
<td>Shift Amt</td>
<td>Mask</td>
<td>XOR</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>Shift</td>
<td>Src Register</td>
<td>Dest Register</td>
<td></td>
<td></td>
<td>Shift Type or Mask</td>
<td>S</td>
<td></td>
</tr>
</tbody>
</table>

#### (e) Floating-point arithmetic instructions

- **A** = Absolute or PC relative
- **L** = Link or subroutine
- **O** = Record overflow in XER
- **R** = Record condition in CR1
- **XO** = Opcode extension
- **S** = Part of shift amount field
- **= 64-bit implementation only**
Foreground Reading

- Stallings chapter 11
- Intel and PowerPC Web sites