Review from Chapter 1: Computer Diagram

- **CPU (Central Processing Unit):** Performs sequences of processing operations on data stored in memory and interacts with Input/Output.
- **Datapath:** Performs basic operation on data stored in registers as directed by Control.
- **Control:** Determines the sequence of data processing operations to be performed in the Datapath.
- **Memory:** An addressable repository for data.
- **Input/Output:** A collection of devices that store, display and convert information.
Control: signals that configure data transfers and establish operations to be performed.

Status: signals that represent the state of data, such as overflow bits, "zero" tests, etc. These signals are tested to change the sequence of operations.

Datapath and Control Unit

- Control: signals that configure data transfers and establish operations to be performed.
- Status: signals that represent the state of data, such as overflow bits, "zero" tests, etc. These signals are tested to change the sequence of operations.
Register Transfer Operations

- **Registers** – a collection of binary storage flip-flops organized in a logical fashion.
- **Register Transfer Operations** – The movement and processing of data stored in registers
- **Three basic components:**
  - set of registers
  - operations
  - control of operations
- **Elementary Operations** -- load, count, shift, add, bitwise "OR", etc.
  - Elementary operations are called *microoperations*

Register Transfer

- **Register Notation**

  - Letters and numbers – denotes a register (ex. R2, PC, IR)
  - Parentheses ( ) – denotes a range of register bits (ex. R1(1), PC(7:0), AR(L))
  - Arrow (→) – denotes data transfer (ex. R1 ← R2, PC(L) ← R0)
  - Comma – separates parallel operations
  - Brackets [ ] – Specifies a memory address (ex. R0 ← M[AR], R3 M[PC])
Conditional Transfer

- If \((K1 = 1)\) then \((R2 \leftarrow R1)\) is shortened to
  \[ K1: (R2 \leftarrow R1) \]
  where \(K1\) is a control variable specifying a conditional execution.

- Conditional execution is used to modify the sequence of microoperations.

Microoperations

- Logical Groupings:
  - Transfer -- move data from one set of registers to another.
  - Arithmetic -- perform arithmetic on data in registers.
  - Logic -- manipulate data or use bitwise logical operations.
  - Shift -- shift data in registers.

<table>
<thead>
<tr>
<th>Arithmetic operations (word-wide)</th>
<th>Logical operations (bitwise)</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ Addition</td>
<td>(\lor) Logical OR</td>
</tr>
<tr>
<td>– Subtraction</td>
<td>(\land) Logical AND</td>
</tr>
<tr>
<td>(*) Multiplication</td>
<td>(\oplus) Logical Exclusive OR</td>
</tr>
<tr>
<td>(/) Division</td>
<td>(\neg) Not</td>
</tr>
</tbody>
</table>
Example Microoperations

- Add the content of R1 to the content of R2 and place the result in R1.
  \[ R1 \leftarrow R1 + R2 \]
- Multiply the content of R1 by the content of R6 and place the result in PC.
  \[ PC \leftarrow R1 * R6 \]
- Exclusive OR the content of R1 with the content of R2 and place the result in R1.
  \[ R1 \leftarrow R1 \oplus R2 \]

Example Microoperations (Continued)

- Take the 1's Complement of the contents of R2 and place it in the PC.
  \[ PC \leftarrow R2 \]
- On condition K1 OR K2, Logical bitwise OR the content of R1 with the content of R3 and place the result in R1.
  \[ (K1 + K2): \ R1 \leftarrow R1 \lor R3 \]
- NOTE: "+" (as in \( K_1 + K_2 \)) and means “OR.” In \( R1 \leftarrow R1 + R3 \), + means “plus.”
Control Expressions

- The control expression for an operation appears to the left of the operation and is separated from it by a colon.
- Control expressions specify the logical conditions for the operation to occur.
- Control expression values of:
  - Logic "1" -- the operation takes place.
  - Logic "0" -- the operation is inhibited.

Examples:
- \( \overline{X} \cdot K1 : R1 \leftarrow R1 + R2 \)
- \( X \cdot K1 : R1 R1 + R2' + 1 \)

Variable \( K1 \) enables the add or subtract operation.
- If \( X = 0 \), then \( \overline{X} = 1 \) so \( \overline{X} \cdot K1 = 1 \), activating the add of \( R1 \) and \( R2 \).
- If \( X = 1 \), then \( X \cdot K1 = 1 \), activating the add of \( R1 \) and the two's comp. of \( R2 \) (subtract).

Arithmetic Microoperations

<table>
<thead>
<tr>
<th>From Table 7-2:</th>
<th>Symbolic Designation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0 ( \leftarrow ) R1 + R2</td>
<td>Addition</td>
<td></td>
</tr>
<tr>
<td>R0 ( \leftarrow ) R1</td>
<td>Ones Complement</td>
<td></td>
</tr>
<tr>
<td>R0 ( \leftarrow ) R1 + 1</td>
<td>Two's Complement</td>
<td></td>
</tr>
<tr>
<td>R0 ( \leftarrow ) R2 + R1 + 1</td>
<td>R2 minus R1 (2's Comp)</td>
<td></td>
</tr>
<tr>
<td>R1 ( \leftarrow ) R1 + 1</td>
<td>Increment (count up)</td>
<td></td>
</tr>
<tr>
<td>R1 ( \leftarrow ) R1 - 1</td>
<td>Decrement (count down)</td>
<td></td>
</tr>
</tbody>
</table>

- Note that any register may be specified for source 1, source 2, or destination.
- These simple microoperations operate on the whole word -- except for 1's complement which is a bitwise operation.
Logical Microoperations

- From Table 7-3:

<table>
<thead>
<tr>
<th>Symbolic Designation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0 ← R1</td>
<td>Bitwise NOT</td>
</tr>
<tr>
<td>R0 ← R1 ∨ R2</td>
<td>Bitwise OR (sets bits)</td>
</tr>
<tr>
<td>R0 ← R1 ∧ R2</td>
<td>Bitwise AND (clears bits)</td>
</tr>
<tr>
<td>R0 ← R1 ⊕ R2</td>
<td>Bitwise EXOR (complements bits)</td>
</tr>
</tbody>
</table>

Let R1 = 10101010, and R2 = 11110000

Then after the operation, R0 becomes:

<table>
<thead>
<tr>
<th>R0</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>01010101</td>
<td>R0 ← R1</td>
</tr>
<tr>
<td>11111010</td>
<td>R0 ← R1 ∨ R2</td>
</tr>
<tr>
<td>10100000</td>
<td>R0 ← R1 ∧ R2</td>
</tr>
<tr>
<td>01011010</td>
<td>R0 ← R1 ⊕ R2</td>
</tr>
</tbody>
</table>

Logical Microoperations (Continued)
Shift Microoperations

- From Table 7-4:
- Let $R_2 = 11001001$
- Then after the operation, $R_1$ becomes:

<table>
<thead>
<tr>
<th>Symbolic Designation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1 \leftarrow \text{sl } R_2$</td>
<td>Shift Left</td>
</tr>
<tr>
<td>$R_1 \leftarrow \text{sr } R_2$</td>
<td>Shift Right</td>
</tr>
</tbody>
</table>

Note: These shifts "zero fill". Sometimes a separate "link" bit can be used to provide the data shifted in, or to "catch" the data shifted out.

Other shifts are possible (circular, arithmetic).

Register Transfer Structures

- **Multiplexer-Based Transfers** - Register inputs are connected to multiple sources via a multiplexer.
- **Bus-Based Transfers** - Register inputs are connected to a single bus driven by a multiplexer.
- **Three-State Bus** - Register inputs and outputs are connected to a single bus via tri-state drivers.
- **Memory Transfer** - Registers provide a source for Memory Addresses and a source or sink for Memory Data.
- **Other Transfer Structures** - Use multiple multiplexers, multiple busses, combinations of all the above, etc.
Multiplexer-Based Transfers

- Multiplexers connected to register inputs produce flexible transfer structures: (Note: Clocks are left off for clarity)

- The transfers are:
  - K1: R0 ← R1
  - K2 K1: R0 ← R2

MUX-Based Transfers (Continued)

- Multiplexers connected to each register input produces a very flexible transfer structure:
- What transfers are possible with this structure? How many operations can occur in parallel?
Bus-Based Transfers

- A single input bus driven by a multiplexer limits the available transfers:
- What transfers can occur here?

Three-State Bus

- The 3-input MUX can be replaced by 3-state buffers. Transfers are still limited:
- What transfers are allowed here?
Memory Transfer

- Memory operations require:
  - ADDRESS
- And require
  - DATA (write operations),
- Or provide
  - DATA (read operations)
- Typically:
  - There can be more than one memory address source in a system.
  - There can be more than one data source or data sink in a system.
  - Some structure of buses and multiplexers is needed to access the memory.

Other Transfer Structures

- Fast systems require that parallel operations occur within the same clock.
- Parallel operations imply "resources" required to move the data.
- SO:
  - Multiple buses are used, and
  - Multiplexers are used to select input sources.
- THIS REQUIRES MORE HARDWARE!
Other Transfer Structures (Continued)

- What transfers does this system allow?