Lecture 23: How does a computer... execute instructions?

Performing Computation

Given topics so far:
Build a sequential circuit for every needed computation
- Hardware designed for specific purpose
- New hardware for every application
- Very impractical
  - Expensive
  - Long time to design and produce

Want general purpose hardware that can execute any program
History Lesson:
Stored Program Computer

1943: ENIAC
• Eckert and Mauchly: first general electronic computer
• Hard-wired program – settings of dials and switches

1944: Beginnings of EDVAC
• Program stored in memory

1945: John von Neumann
• Wrote report on stored program concept

Basic structure became known as “von Neumann machine” (or model)

What do we need to execute general program?

1. Place to store instructions and data
   Memory

2. Perform arithmetic and logical operations
   Processing unit

3. Determine next instruction to execute
   Control unit

4. Get data into computer to manipulate
   Input devices

5. Display results to user
   Output devices
Von Neumann Model

INPUT
Keyboard
Mouse
Scanner
Disk

MEMORY
MAR
MDR

PROCESSING UNIT
ALU
TEMP

OUTPUT
Monitor
Printer
LED
Disk

CONTROL UNIT
PC
IR

datain
control

Random Access Memory

Purpose: Store data and program instructions
Address: Unique ($n$-bit) identifier of $2^n$ locations
Contents: $m$-bit value stored in location

Each variable stored at different address
Each instruction stored at different address

How to move data to/from memory?

Interface defined through two registers + R/W signal
- MAR: Memory Address Register
- MDR: Memory Data Register

Read = Load
Write = Store
Random Access Memory

**How to read Data (kept at location 0011)?**
1. Write address 0011 into MAR
2. Send "read" signal to memory
3. Read data from MDR
   Data is 00101101 = \(32 + 8 + 4 + 1 = 45\)

**How to write value 20 to Points (loc 1101)?**
1. Write data (00010100) to MDR
2. Write address (1101) into MAR.
3. Send "write" signal to memory

---

**Random Access Memory**

**How to change a value in memory?**
- Cannot change memory directly in many instruction sets

**Work with processing unit**
- Load value from memory into register
- Add 1 to value in register
- Write back new value into memory
Processing Unit

Purpose: Manipulate and modify data

1st Component: ALU
- ALU = Arithmetic and Logic Unit
- Many operations
  - ADD, bit-wise AND and NOT
  - Could have many functional units (e.g., multiply, square root)
- Interface: Set one line high (ADD, AND, NOT)

2nd Component: Registers
- Small, temporary storage in addition to memory
- Operands and results of functional units
- Interface: Specify register identified with number

Input and Output

Purpose: Moves data in and out of memory to outside world
- Involves separate hardware device

Some devices provide both input and output
- Disk, network: More on these in later lectures!
Each device has own interface (set of registers)
- Example: Keyboard: data (KBDR) and status (KBSR) registers

Device driver: (part of operating system)
- Low-level software that controls access to device
- Provides common interface to applications
Control Unit

Purpose: Orchestrates execution of program
• Tells processing unit what operations to perform
• Moves data between memory and registers

How does control unit do this?
• Fetches program instructions from memory
• Decodes instructions for processing unit
  ‒ Generates signals telling other components what to do

How does it know what instruction to fetch?
• Program Counter (PC) contains address of next instruction

How does it remember instruction to decode?
• Instruction Register (IR) contains current instruction

Machine Instructions

Definition: Fundamental unit of work
• High-level code is compiled into low-level machine instructions

Instruction specifies two things
• opcode: operation to be performed
• operands: data/locations to be used for operation

Encoded as sequence of bits (just like data!)
• Usually fixed length
• Control unit interprets instruction: generates sequence of control signals to carry out operation

Instruction Set Architecture (ISA)
• Computer’s instructions and formats
Example ISA

Assume: 16 bit instructions
- 16 bit words for data (registers and memory)

Assume: Each instruction has a four-bit opcode
- Bits [15:12] - specification of high-order bits; start with bit 0
  - How many different operations can be performed?
    - 4 bits → $2^4$ combinations = 16 operation

Assume: 8 registers in architecture (R0-R7)
- How many bits needed to specify register?
  - 3 bits → $2^3$ combinations = 8 registers

Explore 3 categories of instructions

Instruction Type #1: Arithmetic and Logical Ops

Example: ADD instruction

What must ADD instruction specify?
- Data: Operand1, Operand2, Result

Where should data reside?
- In registers (too slow to operate on memory)

How to specify location for instruction?
- Each register is numbered; put register number in instruction
Example: ADD Instruction

Assume: Eight registers (R0-R7)

• How many bits are needed to specify each register?
  - 3 bits to specify 8 registers

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>Dst</td>
<td>Src1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Src2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What is this instruction specifying?
“Add the contents of R2 to the contents of R6, and store the result in R6.”

Instruction Type #2: Memory

What operations for dealing with memory?
• Load (read) and Store (write)

What must Load instruction specify?
• Memory address and destination register

Why difficult to put address in instruction?
• Not enough bits
  - 16 bits – 4 bits (opcode) – 3 bits (dest) = 9 bits

How much memory can 9 bits address?
• $2^9 = 512$ words is tiny!!

Put address in register: how much memory can register reach?
• $2^{16} = 64$ K items
• Registers much larger in modern systems!
Example: LDR Instruction

Load: Read data from memory into dest register

Base + offset mode:
- Memory address = Add offset to base register
- Load from memory address into destination register

```
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
LDR  Dst  Base  Offset
```

What is this instruction specifying?
"Add the value 6 to the contents of R3 to form a memory address. Load the contents stored in that address to R2."

Why base+offset useful?

---

Instruction Type #3

Where do we get next instruction from?
- Default: Fetch next sequential instruction

```
when started
move 10 steps
turn 15 degrees
change size by 10
next costume
move 10 steps
turn 15 degrees
change size by 10
next costume
move 10 steps
```

How does Control Unit fetch sequentially?
Set PC = PC + 1
Set MAR = PC
Read from memory
Set IR = MDR
Instruction Type #3: Control

Why does program sometimes want to execute different instruction?
- Program specifies control structure, not straight-line code
- Examples: forever loop, if-then, receive message

Type 3: Instructions that change contents of PC
- Jumps: unconditional (always change PC)
- Branches: conditional (change PC only if some condition is true)

Example: JMP Instruction

Set PC to value contained in register
- Address of next instruction to fetch

<table>
<thead>
<tr>
<th></th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>JMP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Base</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

What is this instruction specifying?
"Load the contents of R3 into the PC."
Today’s Summary

Stored program computers
• Instructions look just like data -- it’s all interpretation
• Three basic kinds of instructions:
  – computational instructions (ADD, AND, …)
  – data movement instructions (LD, ST, …)
  – control instructions (JMP, Branch, …)

Reading
• Pages 188–232 (Chapter 5) of Invitation to CS

Announcements
• Homework 7 due Friday
• Project 2 available