16.070

Introduction to Computers & Programming

Computer Architecture, Machine Language, Program Execution

Prof. Kristina Lundqvist
Dept. of Aero/Astro, MIT
This chapter introduces the activities of a computer's CPU. It describes the machine cycle executed by the control unit and the various operations (or, and, exclusive or, add, shift, etc.) performed by a typical arithmetic/logic unit. The concept of a machine language is presented in terms of the simple yet representative machine described in Appendix C of the text.
Computer architecture =
Interface between Computer and User =
instruction set architecture + computer organization

Computer Architecture

Input
Output
Computer Organization

- **CPU: central processing unit**
  - Part of a computer that controls all the other parts
  - Control Unit: fetches instructions from memory, decodes them and produces signals which control the other parts of the computer.
  - Arithmetic Logic Unit: performs operations such as addition, subtraction, bit-wise AND, OR, …
- **Memory:**
  - Registers, Cache, RAM/ROM
- Temporary buffers
- Logic
Adding values stored in memory

Example 1

**Step 1.** Get one of the values to be added from memory and place it in a register.

**Step 2.** Get the other value to be added from memory and place it in another register.

**Step 3.** Activate the addition circuitry with the registers used in Steps 1 and 2 as inputs and another register designated to hold the result.

**Step 4.** Store the result in memory.

**Step 5.** Stop.
## Levels of Abstraction and Representation

<table>
<thead>
<tr>
<th>High level language program</th>
<th>Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembly language program</td>
<td>Assembler</td>
</tr>
<tr>
<td>Machine language program</td>
<td>ISA</td>
</tr>
<tr>
<td></td>
<td>Machine interpretation</td>
</tr>
<tr>
<td>Control signal spec. or microprogram</td>
<td>Control</td>
</tr>
<tr>
<td>Microprogram execution (circuits)</td>
<td></td>
</tr>
</tbody>
</table>

**Example Code**

```plaintext
Pay := Amount1 + Amount2;
Account := Account - Pay;

LOAD R1,($2)
LOAD R2,($4)
ADD R2,R1,R2
STORE R2,($6)
```

**Machine Code**

```
0000 0011 0011 0100 1101 1110
1001 0001 1011 1101 1100 0100
```

**Control Signals**

- PCout, ARin, READ, DRout, Rlin, PCinc
- ARin, READ

**ALU Operation**
Instruction Set Architecture

- ISA: the parts of a processor’s design that needs to be understood in order to write assembly language
  - Operations (add, sub, mult, …, how is it specified)
  - Number of operands (0, 1, 2, 3)
  - Operand storage (where besides memory)
  - Memory address (how is memory location specified)
  - Type and size of operands (byte, int, float, …)
- Other aspects
  - Parallelism
  - Encoding
  - Successor instruction
  - Conditions
Basic ISA Classes

- Accumulator
- Stack
- General purpose register
- Load/store
Instruction Set

- The collection of machine language instructions that a processor understands
- Instructions are bits with well defined fields
- Instruction format
  - A mapping from instruction to binary values
  - Which bit positions correspond to which parts of the instruction
Instruction Set

- **RISC** (reduced instruction set computer)
  - A processor whose design is based on the rapid execution of a sequence of simple instructions

- **CISC** (complex instruction set computer)
  - Each instruction can perform several low-level operations such as memory access, arithmetic operations or address calculations
The instruction repertoire

- Data transfer
  - LOAD / STORE
  - I/O instructions

- Arithmetic/logic
  - Instructions that tell the CU to request an activity within the ALU (+, -, …, XOR, SHIFT, ROTATE)

- Control
  - Instructions that direct the execution of the program
    - Conditional jumps
    - Unconditional jumps
Dividing values stored in memory

Step 1. LOAD a register with a value from memory.

Step 2. LOAD another register with another value from memory.

Step 3. If this second value is zero, JUMP to Step 6.

Step 4. Divide the contents of the first register by the second register and leave the result in a third register.

Step 5. STORE the contents of the third register in memory.

Step 6. STOP.
The architecture of the machine described in Appendix C
The composition of an instruction for the machine in Appendix C

<table>
<thead>
<tr>
<th>Op-code</th>
<th>0011</th>
<th>Operand</th>
<th>0101 1010 0111</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>A</td>
<td>7</td>
</tr>
</tbody>
</table>

Actual bit pattern (16 bits)

Hexadecimal form (4 digits)
Decoding the instruction 35A7

Op-code 3 means to store the contents of a register in a memory cell.

This part of the operand identifies the address of the memory cell that is to receive data.

This part of the operand identifies the register whose contents are to be stored.
An encoded version of the instructions in Example 1

Example 2

<table>
<thead>
<tr>
<th>Encoded instructions</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>156C</td>
<td>Load register 5 with the bit pattern found in the memory cell at address 6C.</td>
</tr>
<tr>
<td>166D</td>
<td>Load register 6 with the bit pattern found in the memory cell at address 6D.</td>
</tr>
<tr>
<td>5056</td>
<td>Add the contents of register 5 and 6 as though they were two's complement representation and leave the result in register 0.</td>
</tr>
<tr>
<td>306E</td>
<td>Store the contents of register 0 in the memory cell at address 6E.</td>
</tr>
<tr>
<td>C000</td>
<td>Halt.</td>
</tr>
</tbody>
</table>
Example in Ada

X : Integer := 92;
Y : Integer := 90;
Z : Integer;

Z := X + Y;
Example in Machine Language

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
<th>Address</th>
<th>Contents</th>
<th>Address</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0D</td>
<td>B6</td>
<td>14</td>
<td>20</td>
<td>1B</td>
<td>0F</td>
</tr>
<tr>
<td>0E</td>
<td>5C</td>
<td>15</td>
<td>5A</td>
<td>1C</td>
<td>60</td>
</tr>
<tr>
<td>0F</td>
<td>5A</td>
<td>16</td>
<td>30</td>
<td>1D</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>17</td>
<td>0F</td>
<td>1E</td>
<td>30</td>
</tr>
<tr>
<td>11</td>
<td>5C</td>
<td>18</td>
<td>11</td>
<td>1F</td>
<td>0D</td>
</tr>
<tr>
<td>12</td>
<td>30</td>
<td>19</td>
<td>0E</td>
<td>20</td>
<td>C0</td>
</tr>
<tr>
<td>13</td>
<td>0E</td>
<td>1A</td>
<td>12</td>
<td>21</td>
<td>00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Op-code</th>
<th>Operand</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RXY</td>
<td>LOAD RegR from M(XY)</td>
</tr>
<tr>
<td>2</td>
<td>RXY</td>
<td>LOAD RegR with #XY</td>
</tr>
<tr>
<td>3</td>
<td>RXY</td>
<td>STORE M(XY) with RegR</td>
</tr>
<tr>
<td>5</td>
<td>RST</td>
<td>ADDI RegR := RegS + RegT</td>
</tr>
<tr>
<td>C</td>
<td>000</td>
<td>HALT execution</td>
</tr>
<tr>
<td>6</td>
<td>RST</td>
<td>ADDDF RegR := RegS + RegT</td>
</tr>
</tbody>
</table>
The machine cycle

1. Retrieve the next instruction from memory (as indicated by the program counter) and then increment the program counter.

2. Decode the bit pattern in the instruction register.

3. Perform the action requested by the instruction in the instruction register.
Execution Cycle

1. **Instruction fetch**
   - Obtain instruction from program storage

2. **Instruction decode**
   - Determine required actions and instr. size

3. **Operand fetch**
   - Locate and obtain operand data

4. **Execute**
   - Compute results in storage for later

5. **Result store**
   - Deposit results in storage for later use

6. **Next instruction**
   - Determine successor instruction
Decoding the instruction B258

Op-code B means to change the value of the program counter if the contents of the indicated register is the same as that in register 0.

This part of the operand identifies the register to be compared to register 0.

This part of the operand is the address to be placed in the program counter.
Example 2 stored in main memory ready for execution

Program counter contains address of first instructions.

CPU

Main memory

Address     Cells
A0           15
A1           6C
A2           16
A3           6D
A4           50
A5           56
A6           30
A7           6E
A8           CO
A9           00

Program is stored in main memory beginning at address A0.
Performing the fetch step of the machine cycle (continued)

a. At the beginning of the fetch step the instruction starting at address A0 is retrieved from memory and placed in the instruction register.
Performing the fetch step of the machine cycle

**CPU**

- Program counter: A2
- Instruction register: 156C

**Main memory**

<table>
<thead>
<tr>
<th>Address</th>
<th>Cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>15</td>
</tr>
<tr>
<td>A1</td>
<td>6C</td>
</tr>
<tr>
<td>A2</td>
<td>16</td>
</tr>
<tr>
<td>A3</td>
<td>6D</td>
</tr>
</tbody>
</table>

b. Then the program counter is incremented so that it points to the next instruction.
- Pset 2 due tomorrow @ recitation
- Pset 3 on web page today
- Office hours this week
- Lecture Friday: Operating Systems
- Lectures next week: Tony Brogner (Draper)
- ?