Lecture 9 - Programming the Basic Computer: ISA and Assembly Language

Instructor: Bibhas Ghoshal (bibhas.ghoshal@iiita.ac.in)

Spring Semester, 2016
Computer System Structure

The model of a computer can be described by four basic units in high level abstraction. These are:

- Central Processing Unit
- Input Unit
- Output Unit
- Memory
Computer System Structure
So far......

- Structure of Computer components: Logic gates, counters, registers, memory
- Stored program concept - Von Neumann model
- Data representation techniques (integers - positive/negative, floating point numbers)
- Structure of ALU components (for integers and floating point numbers)
What Next???

Instruction Set Architecture

Bibhas Ghoshal
ICOA230C: COA
Spring Semester, 2016

Intel Core i5
What Next????

1. We shall first look at the **software aspect** of the ISA *(assembly programs)*
2. look at implementing the ISA by **designing the processor**

**Instruction Set Architecture - Interface between software and hardware**

- A compiler converts a program into machine instructions in the given ISA
- The processor executes the instructions in the ISA
Write a program in a high level language say in C
Compile it into a format that the computer understands
The compiled code is stored in memory
CPU fetches the compiled code sequentially from memory into its registers
execute the code using Arithmetic and Logic Unit on instructions from the Control Unit
Instructing the Computer

- A Digital Computer is capable of executing instructions
- It can be instructed the sequence of instructions
- User controls the sequence - Program

Program - Set of instructions to specify operations, operands and sequence of occurrence of these instructions

- Instructions are stored in memory (in the form of binary bits) along with data
- The CPU reads them from memory and places them in control registers
- The control register interprets the binary code and proceeds to execute it by issuing a sequence of micro instruction operations
### Instructions

The 16-bit instructions stored in memory
(separate section for program and data)
Machine Language

Programs written to computer can be in any one of the forms:

- **Binary Code**: sequence of instructions and operands in binary that list the exact representation as they appear in computer memory.

- **Hexadecimal code**: equivalent translation of binary to hexadecimal.

- **Symbolic code**: symbols are used for operation part, address part and other parts of the instruction code; each symbolic code is translated to binary using the **Assembler**.

- **High Level Programming language**: special languages developed to reflect procedures used in solution of a problem rather than be concerned about hardware.
### Binary Program to add two numbers

<table>
<thead>
<tr>
<th>Location</th>
<th>Instruction Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0010 0000 0000 0100</td>
</tr>
<tr>
<td>1</td>
<td>0001 0000 0000 0101</td>
</tr>
<tr>
<td>1 0</td>
<td>0011 0000 0000 0110</td>
</tr>
<tr>
<td>1 1</td>
<td>0111 0000 0000 0001</td>
</tr>
<tr>
<td>1 0 0</td>
<td>0000 0000 0101 0011</td>
</tr>
<tr>
<td>1 0 1</td>
<td>1111 1111 1110 1001</td>
</tr>
<tr>
<td>1 1 0</td>
<td>0000 0000 0000 0000</td>
</tr>
</tbody>
</table>

### Hexadecimal Program to add two numbers

<table>
<thead>
<tr>
<th>Location</th>
<th>Instruction Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>2 0 0 4</td>
</tr>
<tr>
<td>0 0 1</td>
<td>1 0 0 5</td>
</tr>
<tr>
<td>0 0 2</td>
<td>3 0 0 6</td>
</tr>
<tr>
<td>0 0 3</td>
<td>7 0 0 1</td>
</tr>
<tr>
<td>0 0 4</td>
<td>0 0 5 3</td>
</tr>
<tr>
<td>0 0 5</td>
<td>F F E 9</td>
</tr>
<tr>
<td>0 0 6</td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>
# Program with Symbolic Codes

<table>
<thead>
<tr>
<th>Location</th>
<th>Instruction Code</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>LDA  004</td>
<td>Load operand 1 in ACC</td>
</tr>
<tr>
<td>0 0 1</td>
<td>ADD  005</td>
<td>Add  operand 2 to ACC</td>
</tr>
<tr>
<td>0 0 2</td>
<td>STA  006</td>
<td>Store sum in 006</td>
</tr>
<tr>
<td>0 0 3</td>
<td>HLT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 0 5 3</td>
<td>First Operand</td>
</tr>
<tr>
<td>0 0 5</td>
<td>F F E 9</td>
<td>Second Operand</td>
</tr>
<tr>
<td>0 0 6</td>
<td>0 0 0 0</td>
<td>Store sum here</td>
</tr>
</tbody>
</table>
The term assembly language refers to a family of low-level programming languages that are specific to an ISA. They have a generic structure that consists of a sequence of assembly statements.

Typically, each assembly statement has two parts:
1. an symbolic instruction code that is a mnemonic for a basic machine instruction
2. list of operands
View of Registers

- Registers: named storage locations\(^1\)
  - ARM - r0,...,r15
  - x86 - eax, ebx, ecx, edx, esi, edi
- Machine specific registers (MSR); examples: control the machine such as speed of fans, power control settings, read on-chip temperature
- Register switch with special functions: program counter, stack pointer, return address

\(^1\)Slide courtesy: Computer Organisation and Architecture by Smruti Sarangi
Memory View

- One large array of bytes
- Each location has an address
- The address of the first location is 0, and increases by 1 for each subsequent location
- The program is stored in a part of the memory
- The program counter contains the address of the current instruction