Instruction Set Architecture

Instructor: Saraju P. Mohanty

PART1

- Computer Architecture Concepts
- Operand Addressing
- Addressing Modes

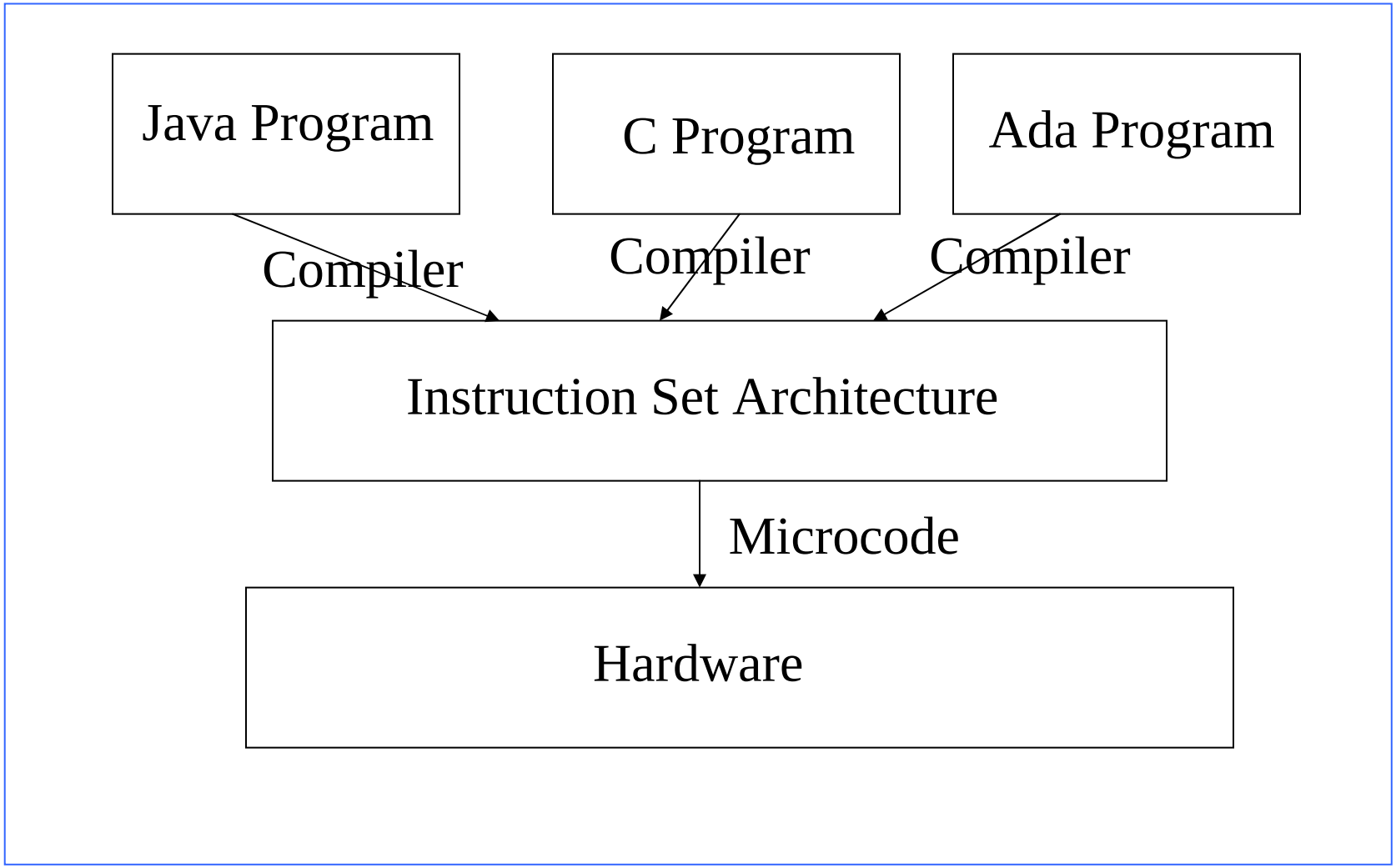
Sources

- Logic and Computer Design Fundamentals by M. M. Mano and C. R. Kime.
- Computer Organization and Design: The Hardware/Software Interface by David A. Patterson and John L. Hennessy.
- Dr. Valavanis lectures

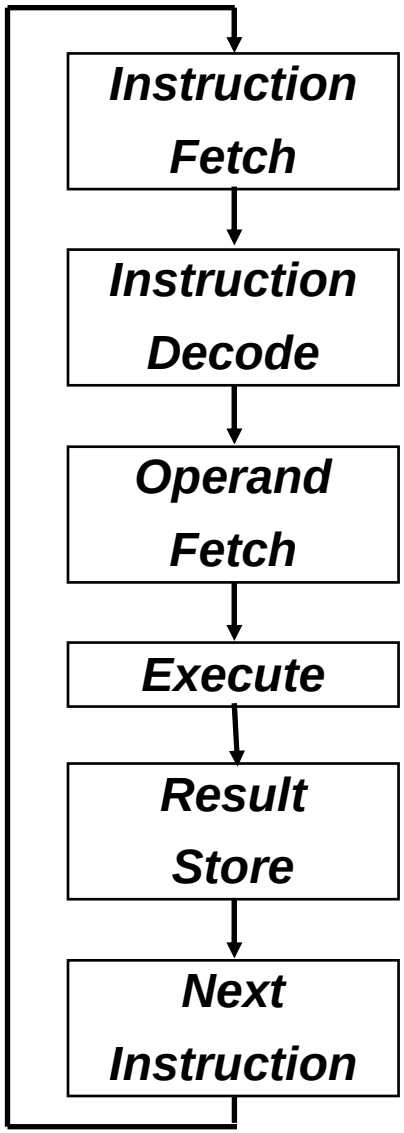
Concepts: Computer Architecture

- Machine Language: The binary language in which instructions are defined and stored in memory.
- Assembly Language: A symbolic language that replaces binary opcodes and addresses with symbolic names and that provides other features helpful to the programmer.
- Computer organization: It consists of structures such as datapaths, control units, memories, and the buses that interconnect them.
- Computer hardware: It refers to the logic, the electronic technologies employed, and the various physical design aspects of the computer.
- Computer architecture: Encompasses the whole computer, including instruction set architecture, organization, and hardware.
- Instruction: A word of the machine language.
- Instruction set architecture: Vocabulary of the machine language.

Concepts: Hardware – Software boundary



Concepts: Basic Computer Operation Cycle



Each instruction is executed in sequence of steps:

- Step 1 : Fetch the instruction from memory into a control register.
- Step 2 : Decode the instruction.
- Step 3 : Locate the operands used by the instruction.
- Step 4 : Fetch operands from memory (if necessary).
- Step 5 : Execute the operation in processor registers.
- Step 6 : Store the results in the proper place.
- Step 7 : Go back to step 1 to fetch the next instruction.

Concepts: What Must be Specified ISA?

- Instruction Format or Encoding
  - how is it decoded?
- Location of operands and result
  - where other than memory?
  - how many explicit operands?
  - how are memory operands located?
  - which can or cannot be in memory?
- Data type and Size
- Operations
  - what are supported
- Successor instruction
  - jumps, conditions, branches

Concepts: Instruction Format

- The instruction format is depicted in a rectangular box symbolizing the bits of the binary instruction.
- Bits are grouped into fields.
- Typical instruction fields:
  - Opcode field – specifies the operation to be performed.
  - Address field – provides either a memory address or an address for selecting a processor register.
  - Mode field – specifies the way the address field is to be interpreted.
- Other additional fields are sometimes used, like an operand field in an immediate operand instruction or a field that gives the number of positions to shift in a shift-type instruction.

Concepts: Register Set

- Register Set consists of all registers in the CPU that are accessible to the programmer.
- Register set includes the programmer-accessible part of the register file and the Program Counter (PC).
- Some of the programmer-accessible registers are,
  - Processor Status Register (PSR) – contains flip-flops that are selectively set by status values C,N,V and Z from the ALU.
  - Stack Pointer (SP).
- The bits of the PSR are referred to as the condition codes or the flags, since they are used to make decisions that determine the program flow.

Stored Program Concept

- Memory can contain the source code, the corresponding complied machine code, and the even the compiler that generates the machine code.
- Both the instructions and data of many types are stored as numbers in memory.
- Thus, it is difficult to determine if the number stored in the memory is an instruction or data !!
- All the memory words that PC points are interpreted as instructions.
- Memory word whose addresses are specified by instructions are interpreted as operands.

Operand Addressing

- Operand residing in memory is specified by its address.
- Operand residing in a processor register is specified by a register address (a binary code of n bits used to specify one among 2^n registers).
- Explicit address – If an operand has an address in the instruction, it is said to be explicitly addressed.
- Implied address – When an operand does not have an explicit address, its location is specified either by the opcode of the instruction or by an address assigned to one of the other operands.
- We will consider (as an example), the ADD instruction that has three operands: the addend, the augend and the result.
- To illustrate the influence of the number of operands on computer programs, we will evaluate the arithmetic statement:

$$X=(A+B)(C+D)$$

Operand Addressing: Three-address instructions

ADD T1, A, B	M[T1] ← M[A]+M[B]
ADD T2, C, D	M[T2] ← M[C]+M[D]
MUL X, T1, T2	M[X] ← M[T1] x M[T2]

M[*] denotes operand at the address symbolized by *. T1, T2 are temporary storage locations in memory.

Same program can use registers as temporary storage locations.

ADD R1, A, B	R1 ← M[A]+M[B]
ADD R2, C, D	R2← M[C]+M[D]
MUL X, R1, R2	M[X] ← R1 * R2

- Advantage: It results in short programs for evaluating expressions.
- Disadvantage: The binary coded instructions require more bits to specify three addresses, particularly if they are memory addresses.

Operand Addressing: Two-address instructions

MOVE T1, A	M[T1] ← M[A]
ADD T1, B	M[T1] ← M[T1]+M[B]
MOVE X, C	M[X] ← M[C]
ADD X, D	M[X] ← M[X]+M[D]
MUL X, T1	M[X] ← M[X] x M[T1]

If temporary storage register R1 available, it can replace T1.

- Each address field can specify either a possible register or a memory address.
- It results in a little longer program than with three-address instructions.

Operand Addressing: One-address instructions

Use an implied address – such as a register called an accumulator ACC for obtaining one of the operands and as the location of the result. So,

LD A	ACC ← M[A]
ADD B	ACC ← ACC+M[B]
ST X	M[X] ← ACC
LD C	ACC ← M[C]
ADD D	ACC ← ACC+M[D]
MUL X	ACC ← ACC * M[X]
ST X	M[X] ← ACC

The number of instructions is more than that with two-address instructions.

Operand Addressing: Zero-address instructions

Use a stack – a structure that stores information such that the item stored last is the first retrieved - last in, first out (LIFO) queue. Word at the top of the stack is referred to as TOS. The one below is as TOS_{-1}. So,

PUSH A	TOS ← M[A]
PUSH B	TOS ← M[B]
ADD	TOS ← TOS+TOS_{-1}
PUSH C	TOS ← M[C]
PUSH D	TOS ← M[D]
ADD	TOS ← TOS=TOS_{-1}
MUL	TOS ← TOS x TOS_{-1}
POP X	M[X] ← TOS

Where TOS is the Top of the Stack. It requires more number of instructions, but, uses no addressed memory locations or registers to execute data manipulation instructions.

Operand Addressing: Addressing Architectures

Addressing architectures are defined by the number of addresses to the memory in the instructions and the number of operands addressed:

- Memory-to-memory: an architecture which has all accesses to memory. e.g. three-address instructions.
- Register-to-register or load/store: allows only one memory address and restricts its use to load and store types of instructions.
- Register-memory: allows one access to memory and one to a register, used primarily to provide compatibility with older software using a specific architecture.
- Single-accumulator architecture: This architecture is used for the one-address instructions and the single address is used for accessing memory.
- Stack architecture: This architecture is used for the zero-address instructions.

Addressing Modes

The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced. The resulting address is called the *effective address*.

Computers use addressing-mode techniques to accommodate one or both of the following provisions:

- Provide programming flexibility to the user via pointers to memory, counters for loop control, indexing of data, and relocation of programs.
- Reduce the number of bits in the address fields of the instruction.

Example of instruction format with a distinct addressing-mode field,

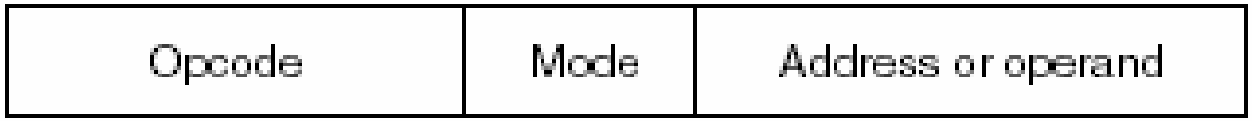


Fig. 9-3 Instruction Format with Mode Field

Addressing Modes: Different Types

- Implied mode – needs no address field at all. The operand is specified implicitly in the definition of the opcode. (e.g.:- accumulator instructions).
- Immediate mode – the operand is specified in the instruction itself. Useful for initializing registers to a constant value.
- Register and register-indirect modes – address field specifies a register. In the indirect mode, the specified register holds the address of the operand in memory.
- Direct addressing mode – the address field of the instruction gives the address of the operand in memory.
- Indirect addressing mode – The address field of the instruction gives the address at which the effective address is stored in memory.
- Relative addressing mode – the address field of the instruction is added to the content of a specified register in the CPU (PC) to evaluate the effective address.

  Effective address = Address part of the instruction + Contents of PC
- Indexed addressing mode – The content of the index register is added to the address part of the instruction to obtain the effective address. The index register may be a special CPU register or simply a register in a register file.