Computer Architecture

Lecture 09

Instruction Sets: Characteristics & Functions

Md. Biplob Hosen
Lecturer, IIT-JU

Reference Books

- Computer Organization and Architecture: Designing for Performance- William Stallings (8th Edition) (Chapter- 10).
  - Any later edition is fine.

Machine Instruction Characteristics

- Instructions that the processor executes.
- The collection of different instructions that the processor can execute is referred to as the processor's instruction set.
- 4 Basic elements of an machine instruction:
  - Operation code
  - Source operand reference
  - Result operand reference
  - Next instruction reference

Elements of a Machine Instruction

- Operation code: Specifies the operation to be performed (e.g., ADD, I/O).
  - The operation is specified by a binary code, known as the operation code, or opcode.
- Source operand reference: The operation may involve one or more source operands, that is, operands that are inputs for the operation.
- Result operand reference: The operation may produce a result.
- Next instruction reference: This tells the processor where to fetch the next instruction after the execution of this instruction is complete.

Elements of a Machine Instruction

- The address of the next instruction could be either:
  - real address or
  - a virtual address
- Depends on architecture.
- Source and result operands can be in one of four areas:
- Main or virtual memory: As with next instruction references, the main or virtual memory address must be supplied.
- Processor register: With rare exceptions, a processor contains one or more registers that may be referenced by machine instructions.
  - If only one register exists, reference to it may be implicit.
  - If more than one register exists, then each register is assigned a unique name or number, and the instruction must contain the number of the desired register.

Elements of a Machine Instruction

- Immediate: The value of the operand is contained in a field in the instruction being executed.
- I/O device: The instruction must specify the I/O module and device for the operation.
  - If memory-mapped I/O is used, this is just another main or virtual memory address.

Instruction Representation

- Each instruction is represented by a sequence of bits, within a computer.
- The instruction is divided into fields, corresponding to the constituent elements of the instruction.
- Opcodes are represented by abbreviations, called *mnemonics, that indicate the* operation.
- Some common examples-

ADD	**Addition**
SUB	Subtraction
DIV	Division
LOAD	Load data from memory
STOR	Store data to memory

Instruction Representation

- Operands are also represented symbolically
- Example: ADD R, Y

Instruction Types

- A typical high level instruction
  - X = X + Y
- How is it done in low level/machine language?
- A computer should have a set of instructions that allows the user to formulate any data processing task
- The set of machine instructions must be sufficient to express any of the instructions from a high-level language
- Basic types-
  - *Arithmetic instructions*
  - *Logic (Boolean) instructions*
  - *memory instructions*
  - *I/O instructions*
  - *Test instructions*
  - *Branch instructions*

Number of Addresses

- Arithmetic and logic instructions will require the most operands.
- Virtually all arithmetic and logic operations are either unary (one source operand) or binary (two source operands).
- Thus, we would need a maximum of two addresses to reference source operands.
- The result of an operation must be stored, suggesting a third address, which defines a destination operand.
- Finally, after completion of an instruction, the next instruction must be fetched, and its address is needed.
- Zero-address instructions are applicable to a special memory organization, called a stack.
- Fewer addresses per instruction result in instructions that are more primitive, requiring a less complex processor.

Number of Addresses

- The design trade-offs involved in choosing the number of addresses per instruction are complicated by other factors.
- There is the issue of whether an address references a memory location or a register.
- Because there are fewer registers, fewer bits are needed for a register reference.
- Also, as we shall see in the next chapter, a machine may offer a variety of addressing modes, and the specification of mode takes one or more bits.
- With one-address instructions, the programmer generally has available only one general-purpose register, the accumulator.
- With multiple-address instructions, it is common to have multiple general purpose registers.

Instruction Set Design

- Most fundamental issues include-
- Operation repertoire: How many and which operations to provide, and how complex operations should be.
- Data types: The various types of data upon which operations are performed.
- Instruction format: Instruction length (in bits), number of addresses, size of various fields, and so on.
- Registers: Number of processor registers that can be referenced by instructions, and their use.
- Addressing: The mode or modes by which the address of an operand is specified.

Types of Operands

- Important general categories are-
  - Addresses
  - Numbers
  - Characters
  - Logical data

Types of Operations

- General types-
  - Data transfer
  - Arithmetic
  - Logical
  - Conversion
  - I/O
  - System control
  - Transfer of control
- Data Transfer: The data transfer instruction must specify several things
  - First, the location of the source and destination operands must be specified.
  - Each location could be memory, a register, or the top of the stack.
  - Second, the length of data to be transferred must be indicated.
  - Third, as with all instructions with operands, the mode of addressing for each operand must be specified.

Types of Operations

- System Control
  - System control instructions are those that can be executed only while the processor is in a certain privileged state or is executing a program in a special privileged area of memory.
  - Typically, these instructions are reserved for the use of the operating system.
- Transfer of Control
- Why required?

1. In the practical use of computers, it is essential to be able to execute each instruction more than once and perhaps many thousands of times.
   - It may require thousands or perhaps millions of instructions to implement an application.
   - This would be unthinkable if each instruction had to be written out separately.
   - If a table or a list of items is to be processed, a program loop is needed.
   - One sequence of instructions is executed repeatedly to process all the data.

Types of Operations

2. Virtually all programs involve some decision making
   - We would like the computer to do one thing if one condition holds, and another thing if another condition holds.
   - For example, a sequence of instructions computes the square root of a number.
   - At the start of the sequence, the sign of the number is tested.
   - If the number is negative, the computation is not performed, but an error condition is reported.
3. To compose correctly a large or even medium-size computer program is an exceedingly difficult task.
   - It helps if there are mechanisms for breaking the task up into smaller pieces that can be worked on one at a time.

- Most common transfer-of-control operations found in instruction sets: branch, skip, and procedure call.

Branch Instructions

- A branch instruction, also called a jump instruction, has as one of its operands the address of the next instruction to be executed.
- Most often, the instruction is a **conditional branch instruction.**
- **That is, the branch is made (update** program counter to equal address specified in operand) only if a certain condition is met.
- Otherwise, the next instruction in sequence is executed (increment program counter as usual).
- A branch instruction in which the branch is always taken is an **unconditional branch.**
- Two ways of doing this:
  - First, most machines provide a 1-bit or multiple-bit condition code that is set as the result of some operations.
  - This code can be thought of as a short user-visible register.
  - BRP X : Branch to location X if result is positive.
  - BRN X :Branch to location X if result is negative.

Branch Instructions

- Another approach that can be used with a three-address instruction format is to perform a comparison and specify a branch in the same instruction.
- For example-
- BRE R1, R2, X :Branch to X if contents of R1 = contents of R2.

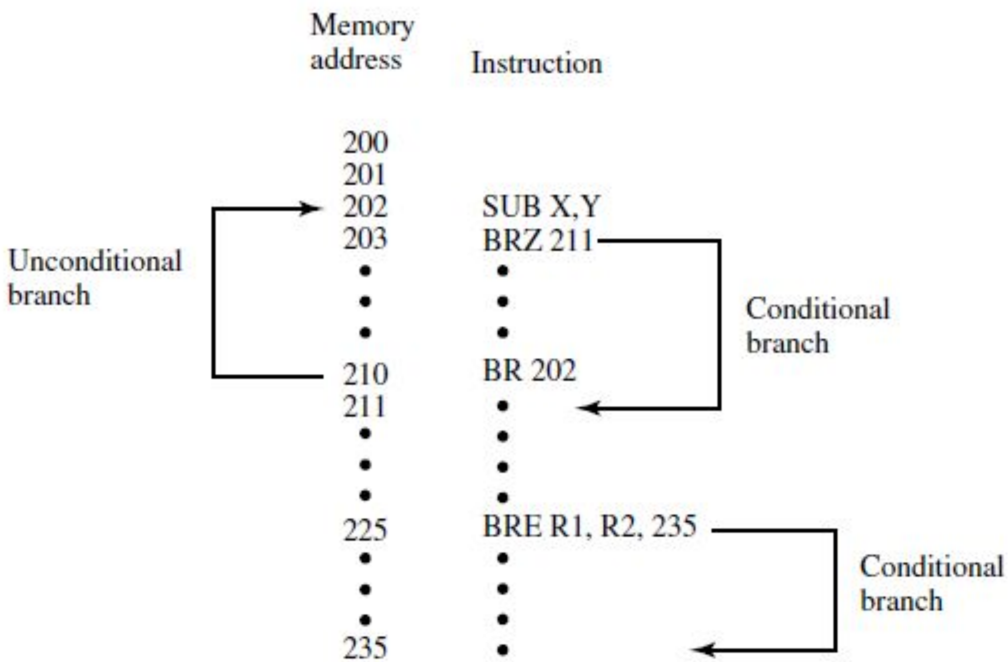

Fig: Branch Instructions

Skip Instructions

- The skip instruction includes an implied address.
- Typically, the skip implies that one instruction be skipped; thus, the implied address equals the address of the next instruction plus one instruction length.
- Because the skip instruction does not require a destination address field, it is free to do other things.
- A typical example is the increment-and-skip-if-zero (ISZ) instruction.

Procedure Call Instructions

- A procedure is a self contained computer program that is incorporated into a larger program.
- At any point in the program the procedure may be invoked, or *called.*
- Used because of-
  - Economy and
  - Modularity
- The procedure mechanism involves two basic instructions:
- A call instruction that branches from the present location to the procedure, and a return instruction that returns from the procedure to the place from which it was called.
  - Both of these are forms of branching instructions.

Procedure Call Instructions

- Important points:
  - A procedure can be called from more than one location.
  - A procedure call can appear in a procedure.
    - This allows the *nesting of procedures* to an arbitrary depth.
  - Each procedure call is matched by a return in the called program.

Thank you!