Computer Archiceture Overview

Presented By

Dr. Banchhanidhi Dash

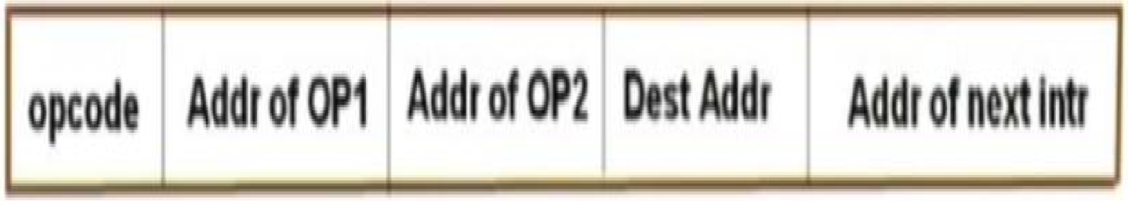
Computer Science &Engineering
KIIT University

outline

- An Instruction
- instruction fomat
- Instruction set Archiecture
- Instruction set Design Issues
- Classifications of ISA
- well known ISA
- RISC vs. CISC

An Instruction

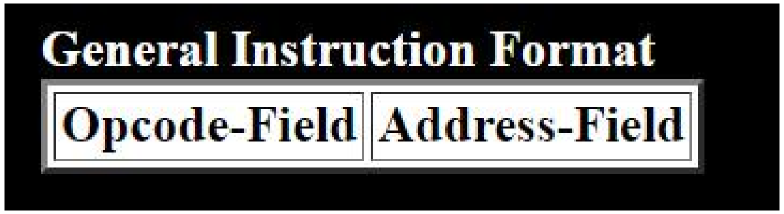
- An instruction ca be treated as a word in a processor language.
- What information an instruction should convey to a CPU??



- The instruction is too long if everything is specified explicitly.
  - more space in memory
  - Longer execution time
- How can you reduce the size of an Instruction??
  - Specifying instruction implicitly.
  - Using PC, Accumulator,GPR,SP

What are Instruction formats?

- An instruction format defines layout of bits of an instruction, in terms of its constituent parts.
- An instruction format must include an opcode and implicitly or explicitly, zero or more operands.
- Each explicit operand is referenced using one of addressing modes.
- Format must, implicitly or explicitly, indicate addressing mode for each operand.
- For most instruction sets, more than on instruction format is used.



Op-field: specifies the operation to be performed;
Address-field: provides operands or the CPU register/MM addresses of the operands.

- An instruction is a single operation of a processor defined by the processor instruction set
- An instruction set is a list of all the commands (instructions), with all their variations, that a processor can execute.
- The physical hardware that is controlled by the instructions is referred to as the Instruction Set Architecture (ISA).
- The encoding of instructions, which is a key aspect of an Instruction Set Architecture, defines how instuctions and arguments are encoded as binary values in the machine code of a system.

Instruction Set Architecture (ISA)

ISA is a structure of a computer that a machine language programmer(or a compiler) must understand to write a correct program for that machine.

The ISA defines

- ✓ the operations that the processor can execute
- ✓ Data transfer mechanism + how to access data
- ✓ control mechanism(branch,Jump,Etc)
- ✓ contract between programmer/compiler and hardware

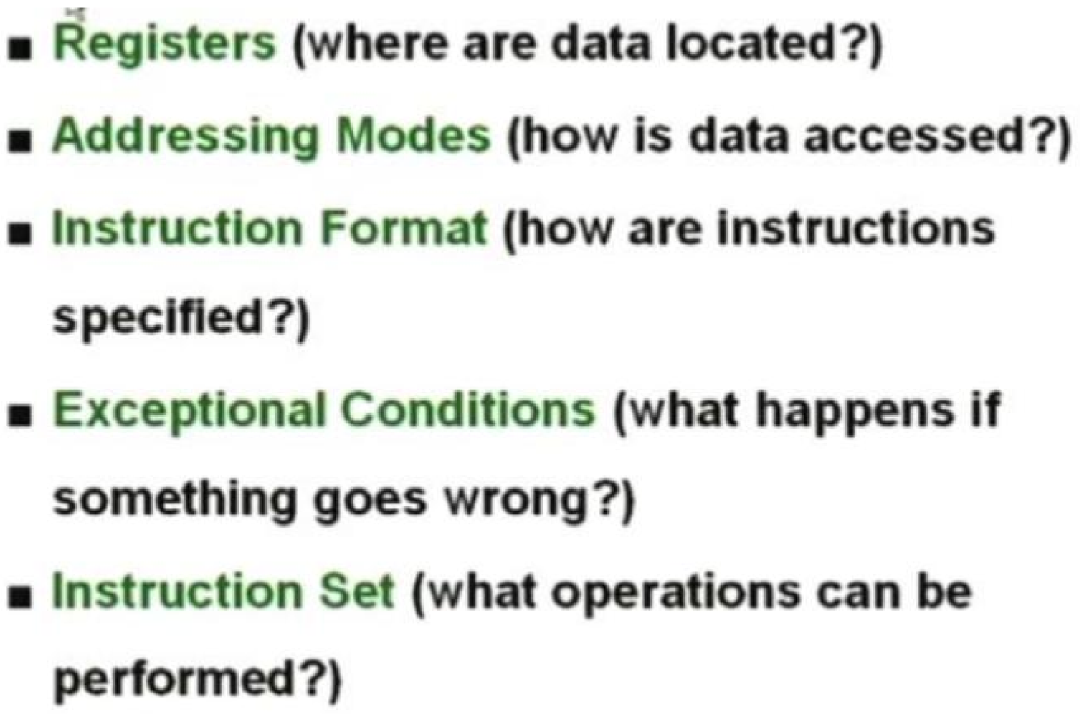
ISA is important.

Not only from programmer prospective

from processor design and implementation prospective.

(ISA)

ISA is programmer visible part of a processor



- Registers (where are data located?)
- Addressing Modes (how is data accessed?)
- Instruction Format (how are instructions specified?)
- Exceptional Conditions (what happens if something goes wrong?)
- Instruction Set (what operations can be performed?)

Instruction Set Design Issues

- Instruction set design issues include:
  - – operand storage in CPU (stack, registers, accumulator)
  - – number of operands in an instruction (fixed or variable number)
  - – type and size of operands
  - – addressing modes,
  - – allowed operations and the size of op-codes,
  - – size of each instruction

Memory addressing

Memory addressing
• Most modern machines are byte-addressable
•• How are bytes addressed within a word?

– Big Endian -- byte 0 is the MSB (IBM, MIPS, SPARC)
– Little Endian -- byte 0 is the LSB (vax, intel 80x86)

Classifying ISAs

Determined by the means used for storing data in CPU

✓ The major choices are:A stack,An accumulator,or a set of registers

Accumulator (before 1960, e.g. 68HC11):

1-address	add A	acc ← acc + mem[A]

Stack (1960s to 1970s):

0-address	add	

Memory-Memory (1970s to 1980s):

2-address	add A, B	mem[A] ← mem[A] + mem[B]
3-address	add A, B, C	mem[A] ← mem[B] + mem[C]

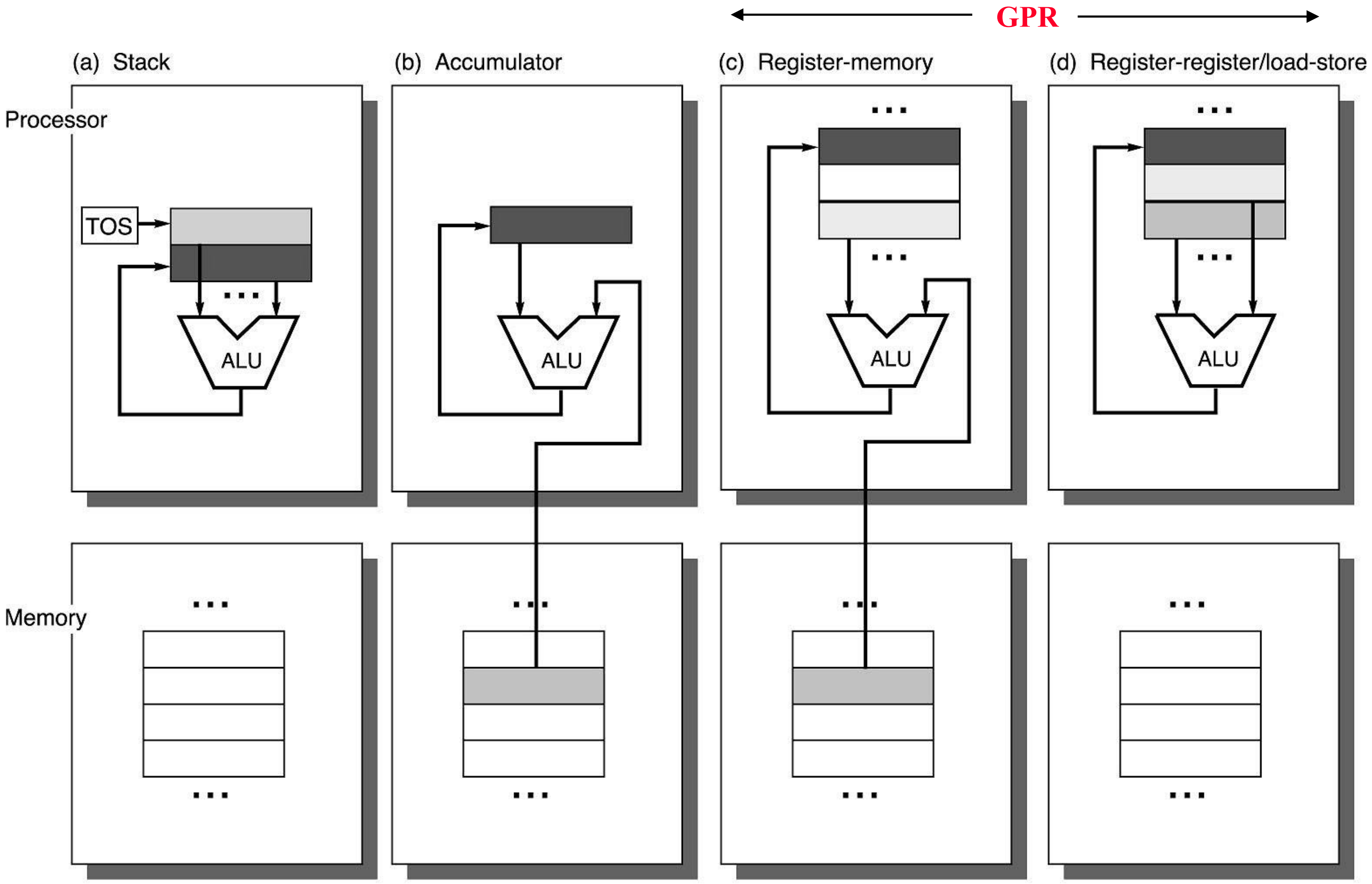
Register-Memory (1970s to present, e.g. 80x86):

2-address	add R1, A	R1 ← R1 + mem[A]
	load R1, A	R1 ← mem[A]

Register-Register (Load/Store) (1960s to present, e.g. MIPS):

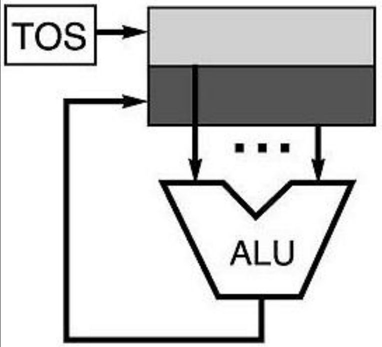
3-address	Load R1,A	R1 ← R1 + mem[A]
	Load R2, B	R2 ← R2 + mem[A]
	Add R3, R1, R2	R3<-R1+R2
	Store R3, C	mem[C]<-r3

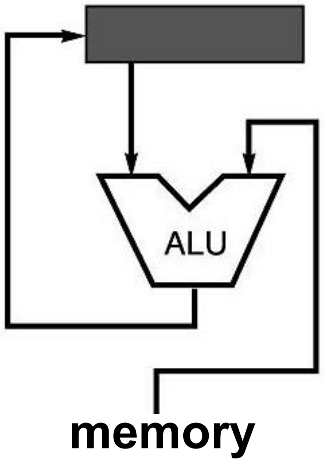
Operand Locations in Four ISA Classes



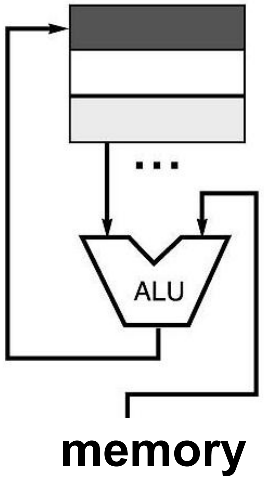
Code Sequence C = A + B for Four Instruction Sets

Stack	Accumulator	Register (register-memory)	Register (load-store)
Push A	Load A	Load R1, A	Load R1,A
Push B	Add B	Add R1, B	Load R2, B
Add	Store C	Store R1, C	Add R3, R1, R2
Pop C			Store R3, C

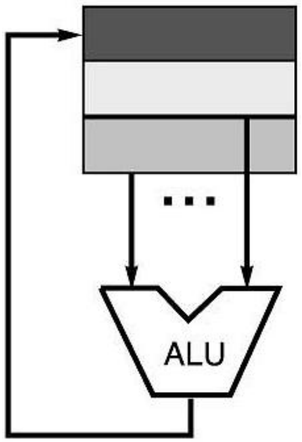




acc = acc + mem[C]



R1 = R1 + mem[C]



R3 = R1 + R2

A load-store architecture is an instruction set architecture that divides instructions into two categories: memory access (load and store between memory and registers) and ALU operations (which only occur between registers).

A register–memory architecture is an instruction set architecture that allows operations to be performed on (or from) memory, as well as registers. If the architecture allows all operands to be in memory or in registers, or in combinations, it is called a "register plus memory" architecture

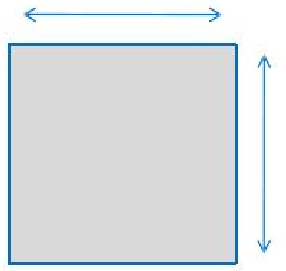
Well Known ISAs

- x86
  - Based on Intel 8086 CPU in 1978
  - CISC machine
  - Intel family, also followed by AMD
  - X86-64
    - 64-bit extensions
    - Proposed by AMD, also followed by Intel
- ARM
  - 32-bit & 64-bit
  - Initially by Acorn RISC Machine
  - ARM Holding
- MIPS
  - 32-bit & 64-bit
  - RISC Machine
  - By Microprocessor without Interlocked Pipeline Stages (MIPS) Technologies

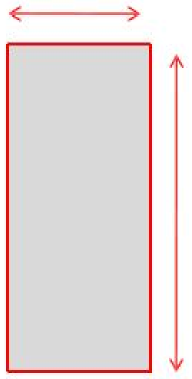
Well Known ISAs (Cont.)

- SPARC
  - 32-bit & 64-bit
  - By Sun Microsystems
  - RISC
- PIC
  - 8-bit to 32-bit
  - By Microchip
- Z80
  - 8-bit
  - By Zilog in 1976
- Many extensions
  - Intel – MMX, SSE(streaming SIMD Extension, SSE2, AVX(Advanced vector extension)

CISC vs. RISC



Code size is smaller but complicated.



Code size is larger but simpler.

CISC vs. RISC

The CISC approach attempts to minimize the number of instructions per program, sacrificing the number of cycles per instruction.

RISC does the opposite, reducing the cycles per instruction at the cost of the number of instructions per program.

CISC and RISC

- ✓ A complex instruction set computer is a computer in which single instructions can execute several low-level operations (such as a load from memory, an arithmetic operation, and a memory store)

- ✓ capable of multi-step operations or addressing modes within single instructions.
- ✓ Examples of CISC architectures
  - ✓ complex mainframe computers to simplistic microcontrollers where memory load and store operations are not separated from arithmetic instructions.
  - ✓ System/360 through z/Architecture, the PDP-11 and VAX architectures.
  - ✓ Well known microprocessors and microcontrollers include the Motorola 6800, 6809 and 68000-families; the Intel 8080, iAPX432 and x86-family;

CISC and RISC

RISC

- ✓ A reduced instruction set computer is a computer with a small, highly optimized set of instructions.
- ✓ the instruction set is optimized with a large number of registers and a highly regular instruction pipeline, allowing a low number of clock cycles per instruction (CPI).
- ✓ Core features of a RISC philosophy are a load/store architecture in which memory is accessed through specific instructions rather than as a part of most instructions in the set, and requiring only single-cycle instructions.

CISC and RISC

RISC

- ✓ RISC designs include ARC, Alpha, Am29000, ARM, Atmel AVR, Blackfin, i860, i960, M88000, MIPS, PA-RISC, Power ISA (including PowerPC), RISC-V, SuperH, and SPARC. The use of ARM architecture processors in smartphones and tablet computers such as the iPad and Android devices provided a wide user base for RISC-based systems.

Characteristics of RISC

- Relatively few instructions.
- Relatively few addressing modes.
- Memory access limited to load and store instructions.
- All operations are done within the registers of the CPU.
- Fixed-length, easily decoded instruction format.
- Single-cycle instruction execution.
- Hardwired rather than microprogrammed control.
- A relatively large number of registers in the processor unit.
- Use of overlapped register windows to speed-up procedure call and return.
- Efficient instruction pipeline.
- Compiler support for efficient translation of high-level language programs into machine language programs.

Characteristics of CISC

- A larger number of instructions – typically from 100 to 250 instructions
- Some instructions that perform specialized tasks and are used infrequently
- A large variety of addressing modes – typically from 5 to 20 different modes
- Variable-length instruction formats
- Instructions that manipulate operands in memory