Computer Architecture, Lecture 3: The processor's structure

Hossam A. H. Fahmy

Cairo University

Electronics and Communications Engineering

Structure versus organization

Architecture: the *structure* and organization of a computer's hardware or system software.

Within the processor: Registers, Operational Units (Integer, Floating Point, special purpose, . . .)

Outside the processor: Memory, I/O, . . .

Instruction Set Architecture (ISA): What is the best for the target application? Why?

Examples: Sun SPARC, MIPS, Intel x86 (IA32), IBM S/390.
Defines: data (types, storage, and addressing modes), instruction (operation code) set, and instruction formats.

ISA

The instruction set is the “language” that allows the hardware and the software to speak.

- It defines the exact functionality of the various operations.
- It defines the manner to perform these operations by specifying the encoding formats.

It does not deal with

- the speed of performing these operations,
- the power consumed when these operations are invoked, and
- the specific details of the circuit implementations.

There are many languages

Arabic and Hebrew are semitic languages. English and Urdu are Indo-European languages. The ISAs have their “families” as well.

Stack architectures push operands onto a stack and issue operations with no explicit operands. The result is put on top of the same stack.

Accumulator architectures use one explicit operand with the accumulator as the second operand. The result is put in the accumulator.

Register-memory architectures have several registers. The operands and the result may be in the registers or the memory.

Register-register architectures have several registers as well. The operands and result are restricted to be in the registers only. *Why?*

Metrics

Code size: Number of instructions in a program and their sizes. It is better to use complex instructions with implicit arguments. Order from best to worst: memory, accumulator, stack, register.

Memory traffic for data: Number of data bytes moved. It is better to have as many operands as possible in the registers. Order: register, stack, accumulator, memory.

CPI: Number of clock cycles to perform the operation. It is better to use simple instructions with little variability. Order: register, stack, accumulator, memory.

Most new architectures are from the load-store type.

CISC versus RISC

Do you remember? The time a program takes is

$$\begin{aligned} \text{CPU time} &= \text{Dynamic instruction count} \\ &\quad \times \text{average CPI} \\ &\quad \times \text{Clock cycle time.} \end{aligned}$$

Complex Instruction Set Computers (CISC) improve the number of instructions.
Reduced Instruction Set Computers (RISC) improve the other two.

What about compatibility?

The longest surviving architectures are the IBM 360/370 and the Intel x86.

- Both are CISC, so why do they survive?
- How are they implemented efficiently to compete with RISC?

Addressing the memory

The (virtual) address space is a critical ISA design decision.

- It affects the encoding formats.
- The usage frequency of the integer data types corresponding to the address size is usually the highest.
- If it is too small, it can limit the lifetime of the ISA.

Memory modes

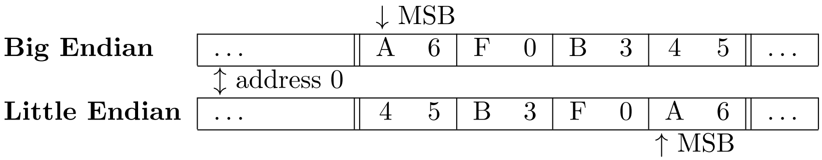
In addition to using registers and getting the operands as immediate constants in the instruction, programs have many ways to specify memory addresses.

Direct:	mem[1204];
Register indirect:	mem[R4];
Displacement:	mem[R1+constant];
Indexed:	mem[R1+R2];
Memory indirect:	mem[mem[R3]];
Autoincrement/decrement:	mem[R3]; R3=R3±constant;
Scaled:	mem[constant1+R2+(R3*constant)];
Relative to PC:	mem[PC+constant];

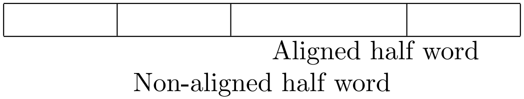
The most widely used are immediate and displacement.

Endian-ness and alignment

How do we represent the number A6F0B345?



Is the hardware capable of accessing any byte or half-word from the memory?



In any case, it should support operations on 8-, 16-, 32-, and 64-bit operands.

Types of instructions

The most widely used (available on all processors) are

- arithmetic and logical,
- data transfer, and
- control instructions.

The following are available on some processors

- system,
- floating point,
- decimal,
- string, and
- graphics instructions.

A look on control instructions

Those instructions include

- conditional branches,
- unconditional jumps,
- function calls, and
- function returns.

They benefit greatly from a PC-relative addressing mode but some cases require register indirect. *Why?*

1. How big is the distance from the program counter?
2. How do we resolve the conditions?

Instruction formats

- Fixed length
  - + Simple memory transfers.
  - + Simple decoding.
  - + Simple pipelines.
  - – Larger code size.
- Variable length.
  - + Compact code.
  - – Hard (but possible) to make the hardware faster.
- Hybrid: two lengths only.
  - + A compromise for embedded systems.

Compiler interaction

For the current compilers technology

- provide a regular and orthogonal instruction set,
    - If done one way somewhere, should be done the same way elsewhere.
    - All the different addressing modes should be independent from the instructions and you should be able to use any mode with any instruction.
- provide primitives not solutions, and
- simplify trade-offs among alternatives.

CPUs, GPUs, and DNNs support backwards compatibility and leverage prior compiler optimizations.