Defining Computer Architecture

The task the computer designer faces is a complex one: Determine what attributes are important for a new computer, then design a computer to maximize performance and energy efficiency while staying within cost, power, and availability constraints. This task has many aspects, including instruction set design, functional organization, logic design, and implementation. The implementation may encompass integrated circuit design, packaging, power, and cooling. Optimizing the design requires familiarity with a very wide range of technologies, from compilers and operating systems to logic design and packaging. Several years ago, the term computer architecture often referred only to instruction set design. Other aspects of computer design were called implementation, often insinuating that implementation is uninteresting or less challenging. We believe this view is incorrect. The architect’s or designer’s job is much more than instruction set design, and the technical hurdles in the other aspects of the project are likely more challenging than those encountered in instruction set design. We’ll quickly review instruction set architecture before describing the larger challenges for the computer architect.

# Instruction Set Architecture

We use the term instruction set architecture (ISA) to refer to the actual programmer-visible instruction set in this book. The ISA serves as the boundary between the software and hardware. This quick review of ISA will use examples from 80x86, ARM, and MIPS to illustrate the seven dimensions of an ISA. Appendices A and K give more details on the three ISAs.

**1. Class of ISA—**Nearly all ISAs today are classified as general-purpose register architectures, where the operands are either registers or memory locations. The 80x86 has 16 general-purpose registers and 16 that can hold floating-point data, while MIPS has 32 general-purpose and 32 floating-point registers (see Figure 1.4 ). The two popular versions of this class are register-memory ISAs, such as the 80x86, which can access memory as part of many instructions, and load-store ISAs, such as ARM and MIPS, which can access memory only with load or store instructions. All recent ISAs are load-store.

**2. Memory addressing—**Virtually all desktop and server computers, including the 80x86, ARM, and MIPS, use byte addressing to access memory operands. Some architectures , like ARM and MIPS, require that objects must be aligned. An access to an object of size s bytes at byte address A is aligned if A mod s = 0. (See Figure A. 5 on page A-8.) The 80x86 does not require alignment, but accesses are generally faster if operands are aligned.

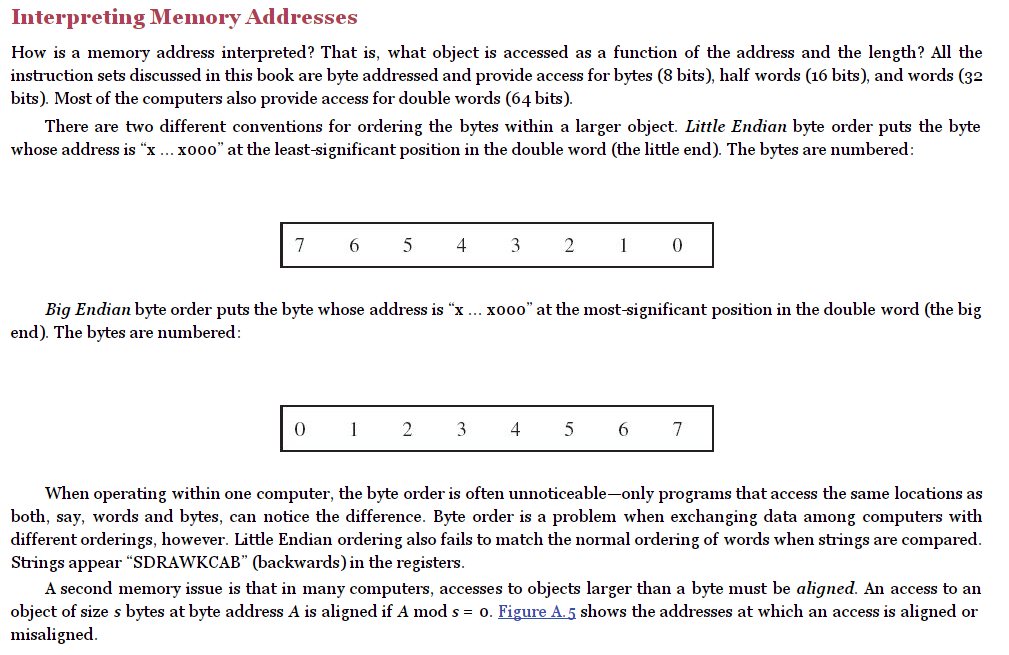
**3. Addressing modes**—In addition to specifying registers and constant operands, addressing modes specify the address of a memory object. MIPS addressing modes are Register, Immediate (for constants), and Displacement, where a constant offset is added to a register to form the memory address. The 80x86 supports those three plus three variations of displacement: no register (absolute), two registers (based indexed with displacement), and two registers where one register is multiplied by the size of the operand in bytes (based with scaled index and displacement). It has more like the last three, minus the displacement field, plus register indirect, indexed, and based with scaled index. ARM has the three MIPS addressing modes plus PC-relative addressing, the sum of two registers, and the sum of two registers where one register is multiplied by the size of the operand in bytes. It also has autoincrement and autodecrement addressing, where the calculated address replaces the contents of one of the registers used in forming the address.

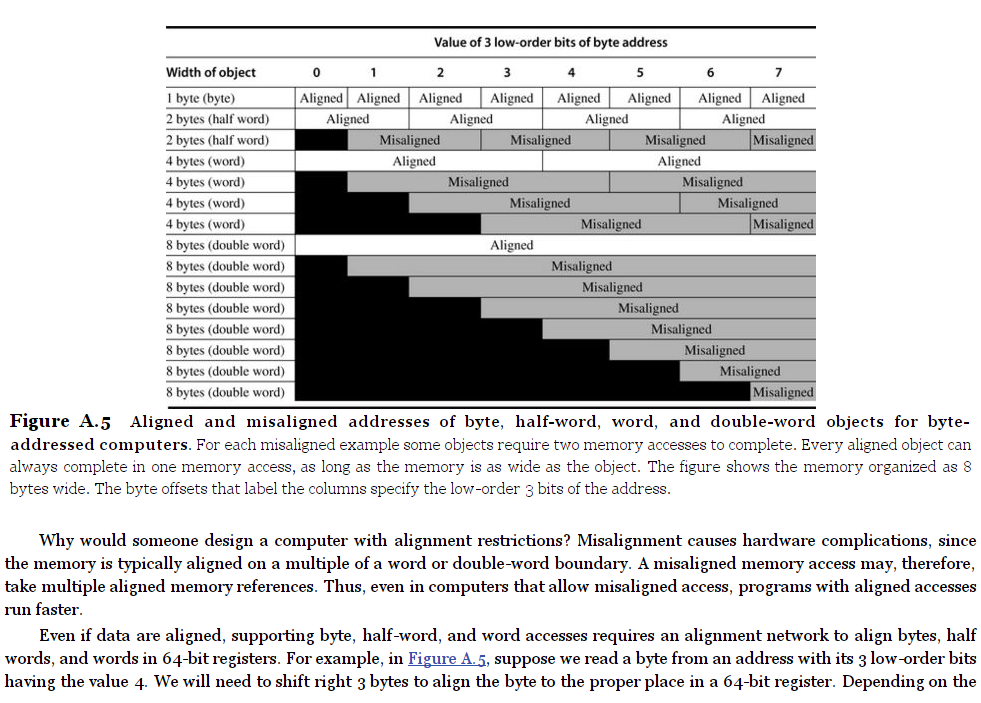
**4. Types and sizes of operands—**Like most ISAs, 80x86, ARM, and MIPS support operand sizes of 8-bit (ASCII character), 16-bit (Unicode character or half word), 32-bit (integer or word), 64-bit (double word or long integer), and IEEE 754 floating point in 32-bit (single precision) and 64-bit (double precision). The 80x86 also supports 80-bit floating point (extended double precision).

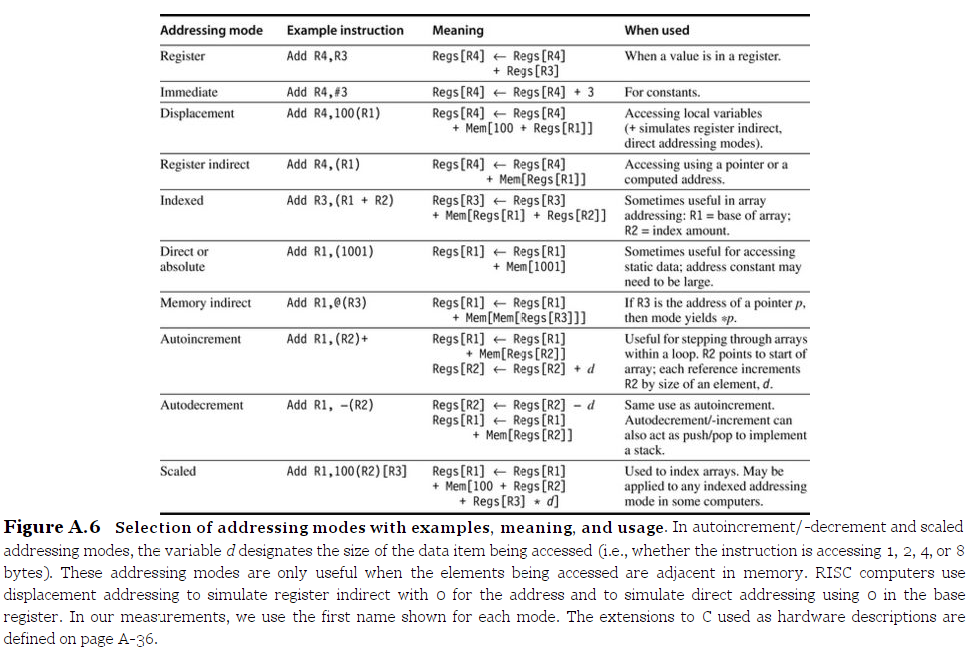
**5. Operations—**The general categories of operations are data transfer, arithmetic logical, control (discussed next), and floating point. MIPS is a simple and easy-to-pipeline instruction set architecture, and it is representative of the RISC architectures being used in 2011. Figure 1.5 summarizes the MIPS ISA. The 80x86 has a much richer and larger set of operations (see Appendix K).

**6. Control flow instructions—**Virtually all ISAs, including these three, support conditional branches, unconditional jumps, procedure calls, and returns. All three use PC-relative addressing, where the branch address is specified by an address field that is added to the PC. There are some small differences. MIPS conditional branches (BE, BNE, etc.) test the contents of registers, while the 80x86 and ARM branches test condition code bits set as side effects of arithmetic/ logic operations. The ARM and MIPS procedure call places the return address in a register, while the 80x86 call (CALLF) places the return address on a stack in memory.

**7. Encoding an ISA—**There are two basic choices on encoding : fixed length and variable length. All ARM and MIPS instructions are 32 bits long, which simplifies instruction decoding. Figure 1.6 shows the MIPS instruction formats. The 80x86 encoding is variable length , ranging from 1 to 18 bytes. Variable-length instructions can take less space than fixed-length instructions, so a program compiled for the 80x86 is usually smaller than the same program compiled for MIPS. Note that choices mentioned above will affect how the instructions are encoded into a binary representation. For example, the number of registers and the number of addressing modes both have a significant impact on the size of instructions, as the register field and addressing mode field can appear many times in a single instruction. (Note that ARM and MIPS later offered extensions to offer 16-bit length instructions so as to reduce program size, called Thumb or Thumb-2 and MIPS16, respectively.)







Hennessy, John L.; Patterson, David A. (2011-10-07). Computer Architecture: A Quantitative Approach (The Morgan Kaufmann Series in Computer Architecture and Design) (Kindle Locations 801-856). Elsevier Science. Kindle Edition.