10.2 Addressing modes and instruction formats

(Original section[1])

The figure below shows the data addressing modes supported by the desktop architectures. Since all have one register that always has the value 0 when used in address modes, the absolute address mode with limited range can be synthesized using zero as the base in displacement addressing. (This register can be changed by ALU operations in PowerPC; it is always 0 in the other machines.) Similarly, register indirect addressing is synthesized by using displacement addressing with an offset of 0. Simplified addressing modes is one distinguishing feature of RISC architectures.

Figure 10.2.1: Summary of data addressing modes supported by the desktop architectures (COD Figure D.2.1).

PA-RISC also has short address versions of the offset addressing modes. MIPS-64 has indexed addressing for floating-point loads and stores. (These addressing modes are described in COD Figure 2.19 (Illustration of four LEGv8 addressing modes).)

Addressing mode	Alpha	MIPS-64	PA-RISC 2.0	PowerPC	SPARCv9
Register + offset (displacement or based)	X	X	X	X	X
Register + register (indexed)		X (FP)	X (Loads)	X	X
Register + scaled register (scaled)			X		
Register + offset and update register			X	X	
Register + register and update register			X	X	

The figure below shows the data addressing modes supported by the embedded architectures. Unlike the desktop RISCs, these embedded machines do not reserve a register to contain 0. Although most have two to three simple addressing modes, ARM and SuperH have several, including fairly complex calculations. ARM has an addressing mode that can shift one register by any amount, add it to the other registers to form the address, and then update one register with this new address.

Figure 10.2.2: Summary of data addressing modes supported by the embedded architectures (COD Figure D.2.2).

SuperH and M32R have separate register indirect and register + offset addressing modes rather than just putting 0 in the offset of the latter mode. This increases the use of 16-bit instructions in the M32R, and it gives a wider set of address modes to different data transfer instructions in SuperH. To get greater addressing range, ARM and Thumb shift the offset left one or two bits if the data size is halfword or word. (These addressing modes are described in COD Figure 2.19 (Illustration of four LEGv8 addressing modes).)

Addressing mode	ARMv4	Thumb	SuperH	M32R	MIPS-16
Register + offset (displacement or based)	X	X	X	X	X
Register + register (indexed)	X	X	X		
Register + scaled register (scaled)	X				
Register + offset and update register	X				
Register + register and update register	X				
Register indirect			X	X	
Autoincrement, autodecrement	X	X	X	X	
PC-relative data	X	X (loads)	X		X (loads)

References to code are normally PC-relative, although jump register indirect is supported for returning from procedures, for *case* statements, and for pointer function calls. One variation is that PC-relative branch addresses are shifted left 2 bits before being added to the PC for the desktop RISCs, thereby increasing the branch distance. This works because the length of all instructions for the desktop RISCs is 32 bits, and instructions must be aligned on 32-bit words in memory. Embedded architectures with 16-bit-long instructions usually shift the PC-relative address by one for similar reasons.

The figure below shows the format of the desktop RISC instructions, which include the size of the address. Each instruction set architecture uses these four primary instruction formats. The subsequent figure shows the six formats for the embedded RISC machines. The desire to have smaller code size via 16-bit instructions leads to more instruction formats.

Figure 10.2.3: Instruction formats for desktop/server RISC architectures (COD Figure D.2.3).

These four formats are found in all five architectures. (The superscrift notation in this figure means the width of a field in bits.) Although the register fields are located in similar pieces of the instruction, be aware that the destination and two source fields are scrambled. Op = the main opcode, Opx = an opcode extension, Rd = the destination register, Rs1 = source register 1, Rs2 = source register 2, and Const = a constant (used as an immediate or as an address). Unlike the other RISCs, Alpha has a format for immediates in arithmetic and logical operations that is different from the data transfer format shown here. It provides an 8-bit immediate in bits 20 to 13 of the RR format, with bits 12 to 5 remaining as an opcode extension.

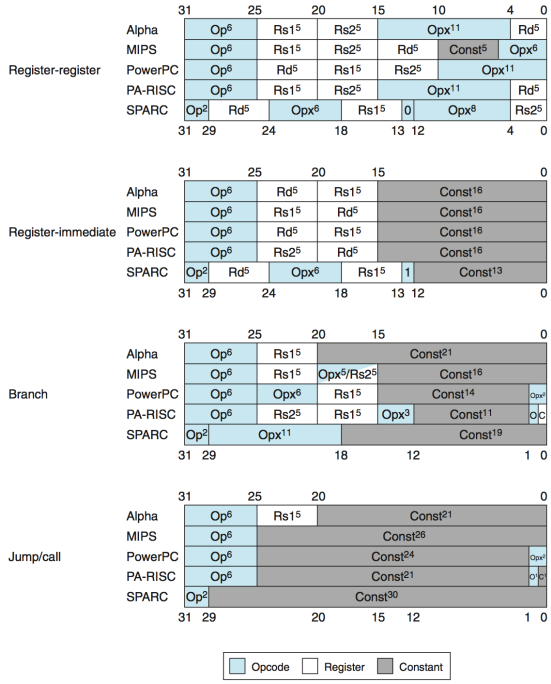
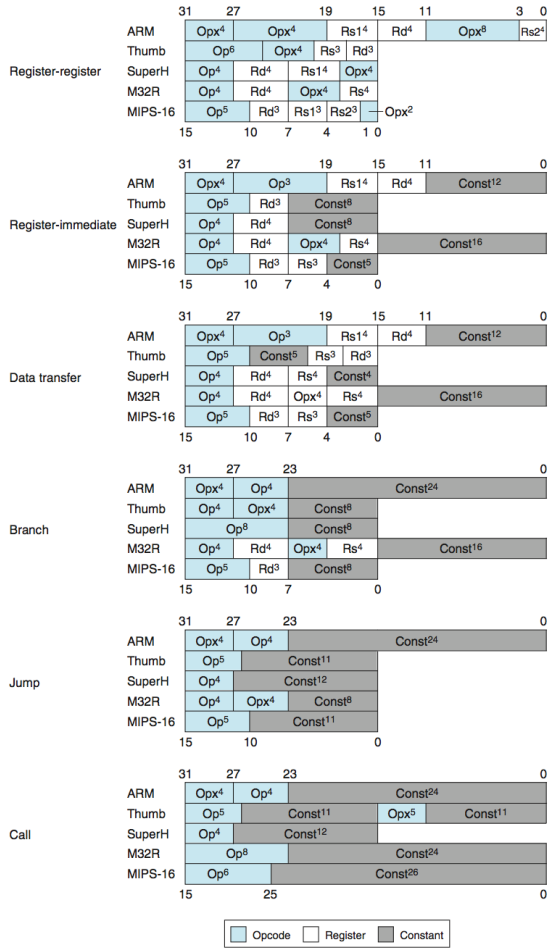


Figure 10.2.4: Instruction formats for embedded RISC architectures (COD Figure D.2.4).

These six formats are found in all five architectures. The notation is the same as in the figure above. Note the similarities in branch, jump, and call formats, and the diversity in register-register, register-immediate, and data transfer formats. The differences result from whether the architecture has eight or 16 registers, whether it is a two- or three-operand format, and whether the instruction length is 16 or 32 bits.



The figures below show the variations in extending constant fields to the full width of the registers. In this subtle point, the RISCs are similar but not identical.

Figure 10.2.5: Summary of constant extension for desktop RISCs (COD Figure D.2.5).

The constants in the jump and call instructions of MIPS are not sign-extended, since they only replace the lower 28 bits of PC, leaving the upper 4 bits unchanged. PA-RISC has no logical immediate instructions.

Format: instruction category	Alpha	MIPS-64	PA-RISC 2.0	PowerPC	SPARCv9
Branch: all	Sign	Sign	Sign	Sign	Sign
Jump/call: all	Sign	—	Sign	Sign	Sign
Register-immediate: data transfer	Sign	Sign	Sign	Sign	Sign
Register-immediate: arithmetic	Zero	Sign	Sign	Sign	Sign
Register-immediate: logical	Zero	Zero	—	Zero	Sign

Figure 10.2.6: Summary of constant extension for embedded RISCs (COD Figure D.2.6).

The 16-bit-length instructions have much shorter immediates than those of the desktop RISCs, typically only 5 to 8 bits. Most embedded RISCs, however, have a way to get a long address for procedure calls from two sequential halfwords. The constants in the jump and call instructions of MIPS are not sign-extended, since they only replace the lower 28 bits of the PC, leaving the upper 4 bits unchanged. The 8-bit immediates in ARM can be rotated right an even number of bits between 2 and 30, yielding a large range of immediate values. For example, all powers of two are immediates in ARM.

Format: instruction category	Armv4	Thumb	SuperH	M32R	MIPS-16
Branch: all	Sign	Sign	Sign	Sign	Sign
Jump/call: all	Sign	Sign/Zero	Sign	Sign	—
Register-immediate: data transfer	Zero	Zero	Zero	Sign	Zero
Register-immediate: arithmetic	Zero	Zero	Sign	Sign	Zero/Sign
Register-immediate: logical	Zero	—	Zero	Zero	—

(*1) This section is in original form.

Provide feedback on this section