PIPELINE AND VECTOR PROCESSING

POTENTIAL PITFALLS IN THE CISC

- The task of the compiler writer is to build a compiler that generates good (fast, small, fast and small) sequences of machine instructions for HLL programs. If there are machine instructions that resemble HLL statements, this task is simplified.
- This reasoning has been disputed by the RISC researchers. They have found that complex machine instructions are often hard to exploit because the compiler must find those cases that exactly fit the construct.
- The task of optimizing the generated code to minimize code size, reduce instruction execution count, and enhance pipelining is much more difficult with a complex instruction set.

POTENTIAL PITFALLS IN THE CISC

- The other major reason cited is the expectation that a CISC will yield smaller, faster programs.
- There are two advantages to smaller programs.
  - As program takes up less memory, there is a savings in that resource. With memory today being so inexpensive, this potential advantage is no longer compelling.
  - Smaller programs should improve performance, and this will happen in three ways. First, fewer instructions means fewer instruction bytes to be fetched. Second, in a paging environment, smaller programs occupy fewer pages, reducing page faults. Third, more instructions fit in cache(s). In many cases, the CISC program, expressed in symbolic machine language, may be *shorter* (i.e., fewer instructions), but the number of bits of memory occupied may not be noticeably *smaller*.
- Because there are more instructions on a CISC, longer opcodes are required, producing longer instructions. Finally, RISCs tend to emphasize register rather than memory references, and the former require fewer bits. An example of this last effect is discussed presently.

Characteristics of Reduced Instruction Set Architecture

- One instruction per machine cycle
  - A *machine cycle* is defined to be the time it takes to fetch two operands from registers, perform an ALU operation, and store the result in a register.
- Register-to-register operations
  - With only simple LOAD and STORE operations accessing memory, this design feature simplifies the instruction set and therefore the control unit.
- Simple addressing modes
- Simple instruction formats

CISC versus RISC Characteristics

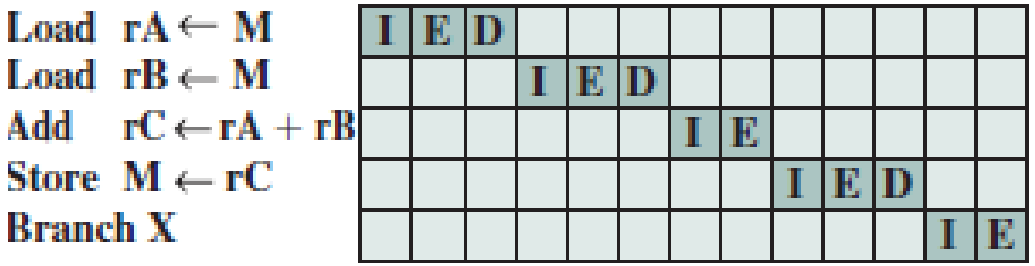
Processor	Number of instruction sizes	Max instruction size in bytes	Number of addressing modes	Indirect addressing	Load/store combined with arithmetic	Max number of memory operands	Unaligned addressing allowed	Max number of MMU uses	Number of bits for integer register specifier	Number of bits for FP register specifier
AMD29000	1	4	1	no	no	1	no	1	8	3[a]
MIPS R2000	1	4	1	no	no	1	no	1	5	4
SPARC	1	4	2	no	no	1	no	1	5	4
MC88000	1	4	3	no	no	1	no	1	5	4
HP PA	1	4	10[a]	no	no	1	no	1	5	4
IBM RT/PC	2[a]	4	1	no	no	1	no	1	4[a]	3[a]
IBM RS/6000	1	4	4	no	no	1	yes	1	5	5
Intel i860	1	4	4	no	no	1	no	1	5	4
IBM 3090	4	8	2[b]	no[b]	yes	2	yes	4	4	2
Intel 80486	12	12	15	no[b]	yes	2	yes	4	3	3
NSC 32016	21	21	23	yes	yes	2	yes	4	3	3
MC68040	11	22	44	yes	yes	2	yes	8	4	3
VAX	56	56	22	yes	yes	6	yes	24	4	0
Clipper	4[a]	8[a]	9[a]	no	no	1	0	2	4[a]	3[a]
Intel 80960	2[a]	8[a]	9[a]	no	no	1	yes[a]	—	5	3[a]

Notes: [a] RISC that does not conform to this characteristic.
[b] CISC that does not conform to this characteristic.

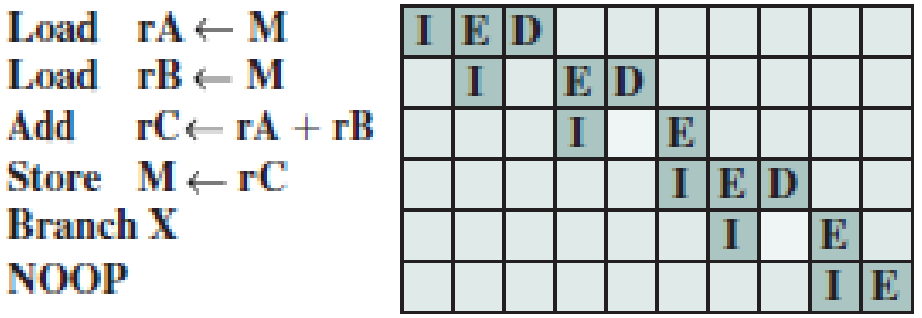
RISC PIPELINING

- Instruction pipelining is often used to enhance performance. In RISC architecture, most instructions are register to register, and an instruction cycle has the following two stages:
  - I : Instruction fetch.
  - E : Execute. Performs an ALU operation with register input and output.
- For load and store operations, three stages are required:
  - I : Instruction fetch.
  - E : Execute. Calculates memory address.
  - D : Memory. Register-to-memory or memory-to-register operation.
- Two problems prevent the maximum speedup
  - only one memory access is possible per stage.
  - branch instruction interrupts the sequential flow of execution. To accommodate this a NOOP instruction can be inserted by the compiler or assembler.
- If an instruction needs an operand that is altered by the preceding instruction, a delay is required. Again, this can be accomplished by a NOOP.

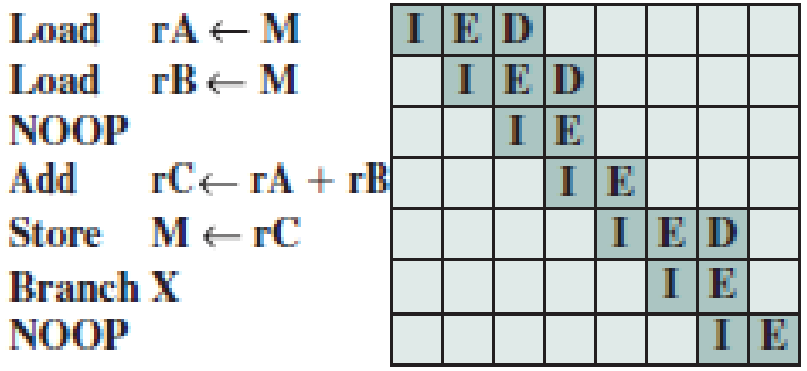
RISC PIPELINING



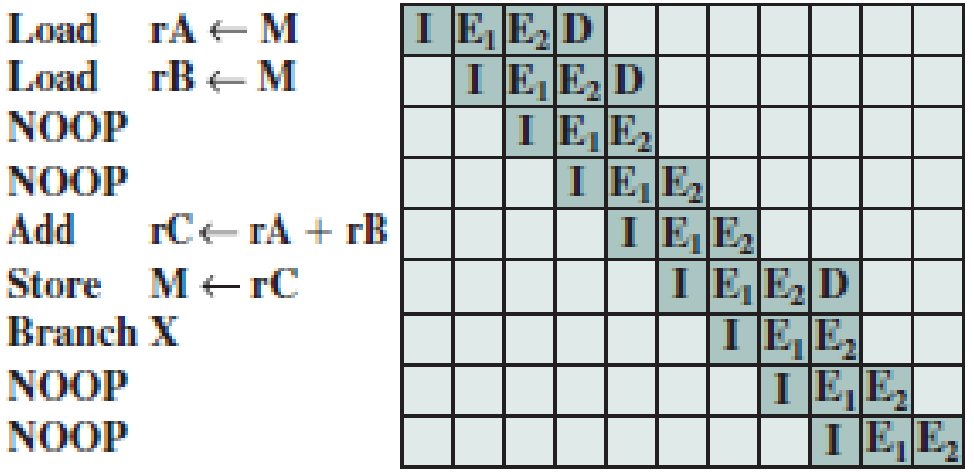
(a) Sequential execution



(b) Two-stage pipelined timing



(c) Three-stage pipelined timing



(d) Four-stage pipelined timing

RISC PIPELINING

- Because the E stage usually involves an ALU operation, it may be longer. In this case, we can divide into two sub-stages:
  - E1: Register read
  - E2: ALU operation and register write
- Because of the simplicity and regularity of a RISC instruction set, the design of the phasing into three or four stages is easily accomplished. Up to four instructions at a time can be under way, and the maximum potential speedup is a factor of 4.

Optimization of Pipelining

- DELAYED BRANCH

Address	Normal Branch		Delayed Branch		Optimized Delayed Branch	
100	LOAD	X, rA	LOAD	X, rA	LOAD	X, rA
101	ADD	1, rA	ADD	1, rA	JUMP	105
102	JUMP	105	JUMP	106	ADD	1, rA
103	ADD	rA, rB	NOOP		ADD	rA, rB
104	SUB	rC, rB	ADD	rA, rB	SUB	rC, rB
105	STORE	rA, Z	SUB	rC, rB	STORE	rA, Z
106			STORE	rA, Z		

Optimization of Pipelining

- DELAYED LOAD
  - On LOAD instructions, the register that is to be the target of the load is locked by the processor. The processor then continues execution of the instruction stream until it reaches an instruction requiring that register, at which point it idles until the load is complete.

Optimization of Pipelining

- LOOP UNROLLING

```
do i=2, n-1
        a[i] = a[i] + a[i-1] * a[i+1]
end do
```

(a) Original loop

```
do i=2, n-2, 2
        a[i] = a[i] + a[i-1] * a[i+1]
        a[i+1] = a[i+1] + a[i] * a[i+2]
end do

if (mod(n-2, 2) = i) then
    a[n-1] = a[n-1] + a[n-2] * a[n]
end if
```

(b) Loop unrolled twice

MIPS - R4000

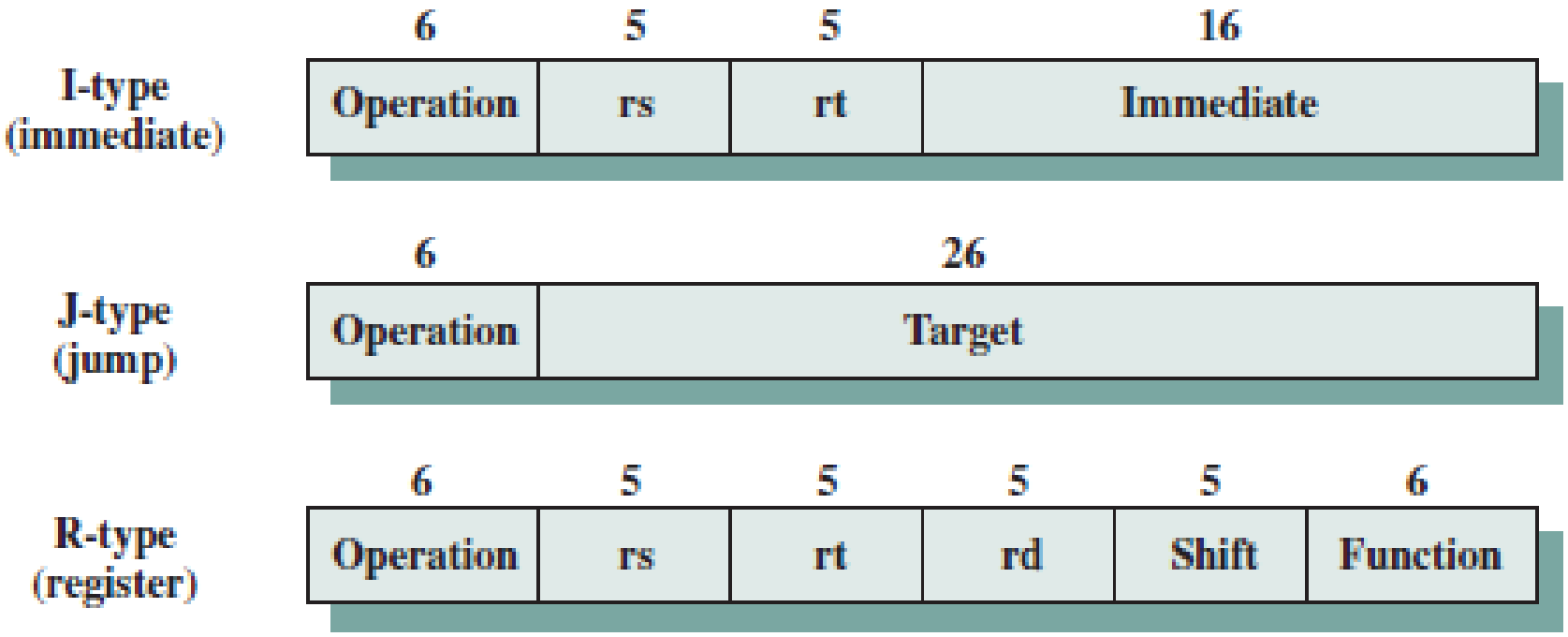
- One of the first commercially available RISC chip sets was developed by MIPS Technology Inc.
- The R4000 uses 64 rather than 32 bits for all internal and external data paths and for addresses, registers, and the ALU.
- It allows a bigger address space large enough for an operating system to map more than a terabyte of files directly into virtual memory for easy access.
- The 64-bit capacity allows the R4000 to process data such as IEEE double-precision floating-point numbers.
- The R4000 processor chip is partitioned into two sections, one containing the CPU and the other containing a coprocessor for memory management.
- The processor supports thirty-two 64-bit registers.
- It also provides for up to 128 Kbytes of high- speed cache, half each for instructions and data. The relatively large cache enables the system to keep large sets of program code and data local to the processor.

MIPS – Instruction Set

- All MIPS R series instructions are encoded in a single 32-bit word format. All data operations are register to register; the only memory references are pure load/store operations.
- As with most RISC-based machines, the MIPS uses a single 32-bit instruction length. The three instruction formats are used.
- Each of the 32 general- purpose registers can be used as the base register.
- The compiler makes use of multiple machine instructions to synthesize typical addressing modes in conventional machines.

MIPS – Instruction Set

- All MIPS R series i
  format. All data o
  references are pu
- As with most RISC-
  instruction length.
- Each of the 32 ge
  register.
- The compiler make
  typical addressing



Operation	Operation code
rs	Source register specifier
rt	Source/destination register specifier
Immediate	Immediate, branch, or address displacement
Target	Jump target address
rd	Destination register specifier
Shift	Shift amount
Function	ALU/shift function specifier

MIPS - Instruction Pipeline

- With its simplified instruction architecture, the MIPS can achieve very efficient pipelining.
- In the R3000, all instructions follow the same sequence of five pipeline stages:
  - Instruction fetch;
  - Source operand fetch from register file;
  - ALU operation or data operand address generation;
  - Data memory reference;
  - Write back into register file.

MIPS - Instruction Pipeline

- The R4000 has eight pipeline stages:
  - **Instruction fetch first half:** Virtual address is presented to the instruction cache and the translation lookaside buffer.
  - **Instruction fetch second half:** Instruction cache outputs the instruction and the TLB generates the physical address.
  - **Register file:** Three activities occur in parallel:
    - — Instruction is decoded and check made for interlock conditions
    - — Instruction cache tag check is made.
    - — Operands are fetched from the register file.
  - **Instruction execute:** One of three activities can occur:
    - — If the instruction is a register-to-register operation, the ALU performs the arithmetic or logical operation.
    - — If the instruction is a load or store, the data virtual address is calculated.
    - — If the instruction is a branch, the branch target virtual address is calculated and branch conditions are checked.
  - **Data cache first:** Virtual address is presented to the data cache and TLB.
  - **Data cache second:** The TLB generates the physical address, and the data cache outputs the data.
  - **Tag check:** Cache tag checks are performed for loads and stores.
  - **Write back:** Instruction result is written back to register file.