The Customer is Always Right

- Compiler is primary customer of ISA
- Features the compiler doesn't use are wasted
- Register allocation is a huge contributor to performance
- Compiler-writer's job is made easier when ISA has
  - regularity
  - primitives, not solutions
  - simple trade-offs
- Summary -> simplicity over power

Okay, putting that all together, what does our desired ISA look like?

Our desired ISA

- Registers, Load-store

Our desired ISA

- Registers, Load-store
- Addressing modes
  - immediate (8-16 bits)
  - displacement (12-16 bits)
  - register deferred (register indirect)

Our desired ISA

- Registers, Load-store
- Addressing modes
  - immediate (8-16 bits)
  - displacement (12-16 bits)
  - register deferred (register indirect)
- Support a reasonable number of operations

Our desired ISA

- Registers, Load-store
- Addressing modes
  - immediate (8-16 bits)
  - displacement (12-16 bits)
  - register deferred (register indirect)
- Support a reasonable number of operations
- Don’t use condition codes

Our desired ISA

- Registers, Load-store
- Addressing modes
  - immediate (8-16 bits)
  - displacement (12-16 bits)
  - register deferred (register indirect)
- Support a reasonable number of operations
- Don’t use condition codes
- Fixed instruction encoding/length for performance

Our desired ISA

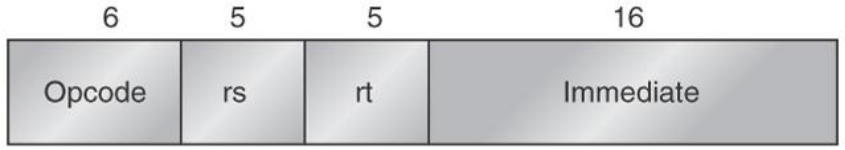
- Registers, Load-store
- Addressing modes
  - immediate (8-16 bits)
  - displacement (12-16 bits)
  - register deferred (register indirect)
- Support a reasonable number of operations
- Don’t use condition codes
- Fixed instruction encoding/length for performance
- Regularity (several general-purpose registers)

MIPS instruction set architecture

- 32 64-bit general-purpose registers (MIPS64 – original MIPS had 32-bit registers)
  - R0 always equals zero
  - 32 FP registers
- immediate and displacement addressing modes
  - register deferred is a subset of displacement
- 32-bit fixed-length instruction encoding

MIPS Instruction Format

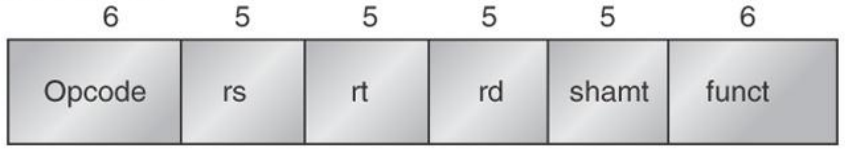
I-type instruction



Encodes: Loads and stores of bytes, half words, words, double words. All immediates (rt ← rs op immediate)

Conditional branch instructions (rs is register, rd unused)
Jump register, jump and link register
(rd = 0, rs = destination, immediate = 0)

R-type instruction



Register-register ALU operations: rd ← rs funct rt
Function encodes the data path operation: Add, Sub, . . .
Read/write special registers and moves

J-type instruction

Jump and jump and link
Trap and return from exception

Point out regularity, also point out Example instructions.

Category	Instr	Op Code	Example	Meaning
Arithmetic (R format)	add	0 and 32	add $s1, $s2, $s3	$s1 = $s2 + $s3
	subtract	0 and 34	sub $s1, $s2, $s3	$s1 = $s2 - $s3
Data transfer (I format)	load word	35	lw $s1, 100($s2)	$s1 = Memory($s2+100)
	store word	43	sw $s1, 100($s2)	Memory($s2+100) = $s1
	load byte	32	lb $s1, 101($s2)	$s1 = Memory($s2+101)
	store byte	40	sb $s1, 101($s2)	Memory($s2+101) = $s1
Cond. Branch	br on equal	4	beq $s1, $s2, L	if ($s1==$s2) go to L
	br on not equal	5	bne $s1, $s2, L	if ($s1 !=$s2) go to L
	set on less than	0 and 42	slt $s1, $s2, $s3	if ($s2<$s3) $s1=1 else $s1=0
Uncond. Jump	jump	2	j 2500	go to 10000
	jump register	0 and 8	jr $t1	go to $t1
	jump and link	3	jal 2500	go to 10000; $ra=PC+4

$ra is the Return Address register, which is R31

A few sample instructions

lw R1, 1000(R2)

dadd R1, R2, R3

daddi R1, R2, #53

jal label

jr R3

beq R1, R5, label

RTL

RISC vs CISC

- MIPS is a classic RISC architectures (as are SPARC, Alpha, PowerPC, …)
- RISC stands for Reduced Instruction Set Computer. RISC architectures are load-store, few formats, minimal instruction sets.
- They were in contrast to the 70s and 80s which proliferated CISC ISAs (VAX, Intel x86, various IBM), which were characterized by complex and comprehensive instruction sets, and complex instruction decoding.
- RISC architectures thrived not because they supported fewer operations, but because they *enabled parallelism*.

MIPS R2000 vs. VAX 8700

Or “Why RISC?”

$$ET = IC * CPI * CT$$

Punch line: $CPI_{VAX} = 6\ CPI_{MIPS}$

$$IC_{MIPS} = 2\ IC_{VAX}$$

MIPS R2000, first MIPS in Jan 1986 and executed MIPS 1. First to have 5-stage RISC pipeline. 80mm, Unfair to compare clock speed 15MHZ, 66ns

VAX 8700 also 1986, 32 bit, also 5-stage processor, 45ns CT

If curious:

Table 1: Machine Implementation Parameters

	VAX 4000/300	MIPS M/2000	VAX 8700
Chip First Silicon	1989	1988	n/a
System Ship	1990	1989	1986
CPU	REX520	R3000	n/a
Technology	Custom CMOS	Custom CMOS	ECL gate array
Component counts			
CPU	140K transistors, 180Kbits mem	115K transistors	approx. 100 gate arrays, 1200 gates each
FPU	134K transistors	105K transistors	(included above)
Feature size	1.5 micron	1.2 micron	
Die size			
CPU	12x12 mm^2	7.6x8.7 mm^2	n/a
FPU	12.7x11 mm^2	12.6x12.6 mm^2	
Cycle time	28 ns.	40 ns.	45 ns.
On-chip cache	2 KB	none	n/a
Board cache	128 KB I+D	64 KB I, 64 KB D	64 KB I+D
TLB	64 entries	64 entries	1024 entries
Page size	512 bytes	4 Kbytes	512 bytes
Memory access time	13 cycles	12 cycles	16 cycles
FP multiply	15 cycles	5 cycles	15 cycles
FP Add	14 cycles	2 cycles	11 cycles
List price	$100K	$80K	$492K
Performance			
Overall SPECmark	7.9	17.6	5.6
Integer SPECmark	7.7	19.7	5.0
FP SPECmark	8.1	16.3	6.0

Bhandarkar and Clark. Performance from Architecture: Comparing a RISC and a CISC with Similar Hardware Organization. ASPLOS 1991.

ISA Key Points

- Modern ISA's typically sacrifice power and flexibility for regularity and simplicity; code density for parallelism and throughput.

ISA Key Points

- Modern ISA's typically sacrifice power and flexibility for regularity and simplicity; code density for parallelism and throughput.
- instruction bits are extremely limited, particularly in a fixed-length instruction format.

ISA Key Points

- Modern ISA's typically sacrifice power and flexibility for regularity and simplicity; code density for parallelism and throughput.
- instruction bits are extremely limited, particularly in a fixed-length instruction format.
- Registers are critical to performance – we want lots of them, and few strings attached.

ISA Key Points

- Modern ISA's typically sacrifice power and flexibility for regularity and simplicity; code density for parallelism and throughput.
- instruction bits are extremely limited, particularly in a fixed-length instruction format.
- Registers are critical to performance – we want lots of them, and few strings attached.
- Displacement addressing mode handles the vast majority of memory reference needs.