RISC (MIPS, RISC-V) and CISC (x86) Instruction Sets

Jaynarayan T Tudu

Lecture 6

IIT Tirupati, India

13th Feb, 2020

Computer System Architecture

Review From Last Lecture

- Introduction and recent trends in Computer Architecture
- Fundamentals of Computer System Design
- Measuring Performances: Benchmark and Metric
- Analysis of Instructions
- Addressing modes

Reduced Instruction Set Computer Architecture

Important features of RISC architecture.

- Simple instructions
- Fixed Instruction Encoding
- Simple controller
- Load/Store architecture (is this good or bad?)
- Instruction count increases
- Limited addressing modes

RISC and Its Variants

Desktop and Server:

- Digital Alpha
- MIPS, Inc.
- Hewlett-Packard PA-RISC
- IBM and Motorola PowerPC
- Sun Microsystems SPARC

Embedded Computing and Mobile Devices:

- Advanced RISC Machines ARM
- Advanced RISC Machines Thumb
- Hitachi SuperH
- Mitsubishi M32R
- MIPS, Inc. MIPS16

The MIPS Architecture

The emphasis of MIPS architecture:

- A simple load-store instruction set
- Designed for pipe-line efficiency
- Fixed instruction encoding
- Efficiency as compiler target

Register for MIPS (MIPS64)

Classification of registers:

- General purpose registers (Integer registers)
- Floating point registers
- Special purpose registers

General purpose registers also known as Integer register:
Size 64 bits

63	0
R0	
R1	
......	
R31	

Register for MIPS (MIPS64)

Classification of registers:

- General purpose registers (Integer registers)
- Floating point registers
- Special purpose registers

Floating point register:
Size 64 bits: it support single and double-precision

63	0
F0	
F1	
......	
F31	

Register for MIPS (MIPS64)

Usage of registers

R0	Hard-wired to 0
R1	Reserved for pseudo-instructions
R2 - R3	Return values from functions
R4 - R7	Arguments to functions - not preserved by subprograms
R8 - R15	Temporary data, not preserved by subprograms
R16 - R23	Saved registers, preserved by subprograms
R24 - R25	More temporary registers, not preserved by subprograms
R26 - R27	Reserved for kernel. Do not use.
R28	Global Area Pointer (base of global data segment)
R29	Stack Pointer
R30	Frame Pointer
R31	Return Address
F0 - F3	Floating point return values
F4 - F10	Temporary registers, not preserved by subprograms
F12 - F14	First two arguments to subprograms, not preserved by subprograms
F16 - F18	More temporary registers, not preserved by subprograms
F20 - F31	Saved registers, preserved by subprograms

Data Type for MIPS (MIPS64)

Integer:

- 8-bit (byte)
- 16-bit (half word)
- 32-bit (word)
- 64-bit (double word)

Floating-point:

- 32 bit single precision
- 64-bit double precision

Addressing Modes for MIPS

MIPS64 addressing mode for data transfer:

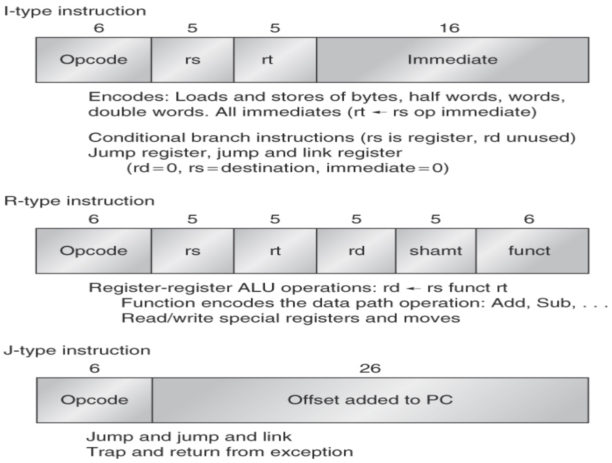
- It has only three type of addressing modes
- I - type: Immediate
- R - type: Register-register
- J - type: Jump (Offset added to PC, index mode)

However indirectly supports:

- Displacement mode
- Register indirect
- Absolute addressing

Addressing Modes for MIPS

Three type of addressing modes with fixed operands:



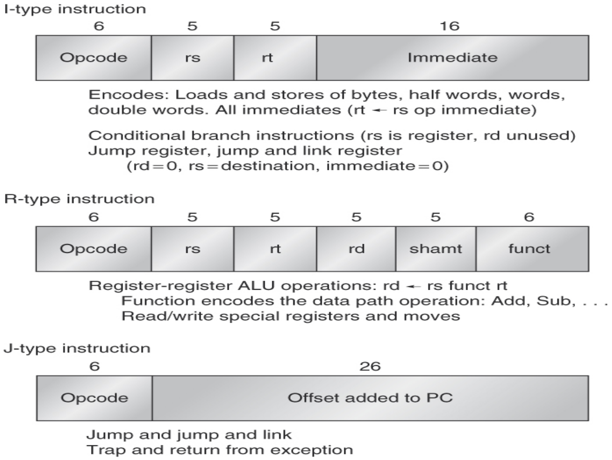
I-type instruction
6
5
5
16
Opcode
rs
rt
Immediate
Encodes: Loads and stores of bytes, half words, words, double words. All immediates (rt ← rs op immediate)
Conditional branch instructions (rs is register, rd unused)
Jump register, jump and link register
(rd=0, rs=destination, immediate=0)
R-type instruction
6
5
5
5
5
6
Opcode
rs
rt
rd
shamt
funct
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
6
26
Opcode
Offset added to PC
Jump and jump and link
Trap and return from exception

Addressing Modes for MIPS

With respect to Memory:

- Byte addressable
- 64 bit address space
- Mode bit for Big and Little support (this is good)
- All the memory access must be aligned (why?)

Instruction Format for MIPS


I-type instruction
6
5
5
16
Opcode
rs
rt
Immediate
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
6
5
5
5
5
6
Opcode
rs
rt
rd
shamt
funct
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
6
26
Opcode
Offset added to PC
Jump and jump and link
Trap and return from exception

Instructions are of 32 bit length.

Operations in MIPS

Broadly operations are classified in four:

1. Load and Store operations
2. ALU operations
3. Branch and Jump operations
4. Floating point operations

Operations in MIPS: Loads and Stores

LD R1, 30(R2)	Load double word
LD R1, 1000(R0)	Load double word
LW R1, 60(R2)	Load word
LB R1, 40(R3)	Load byte
LBU R1, 40(R3)	Load byte unsigned
L.S F0, 50(R3)	Load FP single
L.D F0, 50(R2)	Load FP double
SD R3, 500(R4)	Store double word
SW R3, 500(R4)	Store word
SH R3, 502(R2)	Store half
SB R2, 41(R3)	Store byte
S.S F0, 40(R3)	Store FP single
S.D F0, 40(R3)	Store FP double

Operations in MIPS: Arithmetic Instructions

DADDU R1, R2, R3	Add unsigned
DADDIU R1, R2, #3	Add immediate unsigned
LUI R1, #42	Load upper immediate
DSLL R1, R2, #5	shift left logical
SLT R1, R2, R3	Set less than

- Instruction with Immediate Operands: ADDI, ADDI ADDIU, ANDI, LUI, ORI, SLTI, SLTIU, XORI
- Three operand instructions: ADD, ADDU, AND, NOR, OR, SLT, SLTU, SUB, SUBU, XOR
- ALU Two operand Instruction: CLO, CLZ, NOR, OR, XOR
- Shift instructions: SLL, SLLV, SRA, SRAV, SRL, SRLV
- Mutiply and Divide: DIV, DIVU, MADD, MADDU, MFHI, MFLO, MSUB, MSUBU, MTHI, MTLO, MUL, MULT, MULTU

Operations in MIPS: Control flow Instructions

- J name $PC_{36......63} \leftarrow$ name
- JAL name Regs[R31] $\leftarrow$ PC + 8; $PC_{36...63} \leftarrow$ name
- JALR R2 Regs[R31] $\leftarrow$ PC + 8; PC $\leftarrow$ Reg[R2]
- JR R3 PC $\leftarrow$ Reg[R3]
- BEQZ R4, name if R4 == 0 then PC $\leftarrow$ name
- BNE R3, R4, name if R3 != R4 then PC $\leftarrow$ name
- MOVZ R1, R2, R3 if R3 == 0 then R1 $\leftarrow$ R2

Operations in MIPS: Floating Point

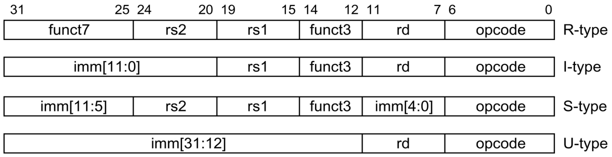
The principle:

- Manipulates the floating point registers
- Indicate the operation to be single or double precession

- FP addition: ADD.S and ADD.D
- FP subtraction: SUB.S and SUB.D
- FP multiplication: MUL.S and MUL.D
- FP division: DIV.S and DIV.D
- FP comparison: C.X.S and C.X.D
- FP branch true:BCLT and branch false: BCLF

RISC-V Instruction Set Architecture

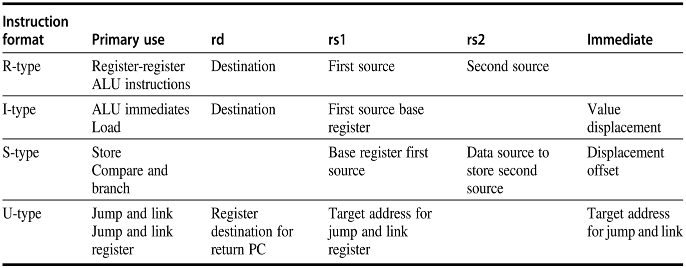
Instruction set format:


31
25 24
20 19
15 14
12 11
7 6
0
funct7 | rs2 | rs1 | funct3 | rd | opcode
R-type
imm[11:0] | rs1 | funct3 | rd | opcode
I-type
imm[11:5] | rs2 | rs1 | funct3 | imm[4:0] | opcode
S-type
imm[31:12] | rd | opcode
U-type

The RISC-V has four different format of fixed length. This is very similar to MIPS where it has three isntruction format.
Valid question: Why did RISC-V chose additional instruction format?

RISC-V Instruction Set Architecture

Description of instruction set format:



Instruction format	Primary use	rd	rs1	rs2	Immediate
R-type	Register-register ALU instructions	Destination	First source	Second source	
I-type	ALU immediates Load	Destination	First source base register		Value displacement
S-type	Store Compare and branch		Base register first source	Data source to store second source	Displacement offset
U-type	Jump and link Jump and link register	Register destination for return PC	Target address for jump and link register		Target address for jump and link

The RISC-V has four different format of fixed length. This is very similar to MIPS where it has three isntruction format.
Valid question: Why did RISC-V chose additional instruction format?

RISC-V Instruction Set Architecture

The riscv classify instructions into four classes:

- Load/store
- Control
- ALU

Reading Material

- Appendix A of text book: Henessy and Patternson, Computer Architecture Quatitative Approach, 5th Edition
- Appendix K of text book: Available online.
- Relevant Papers on MIPS32 and MIPS64

thank you