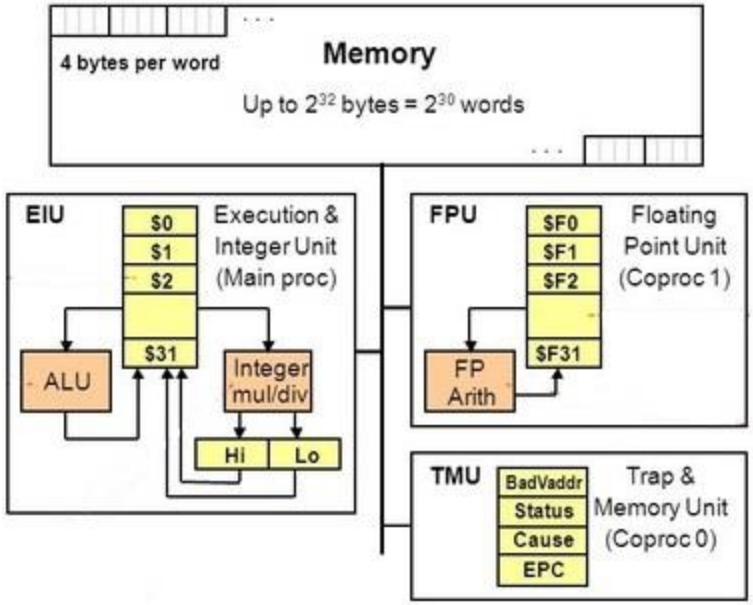
معماری کامپیوتر

فصل پنج

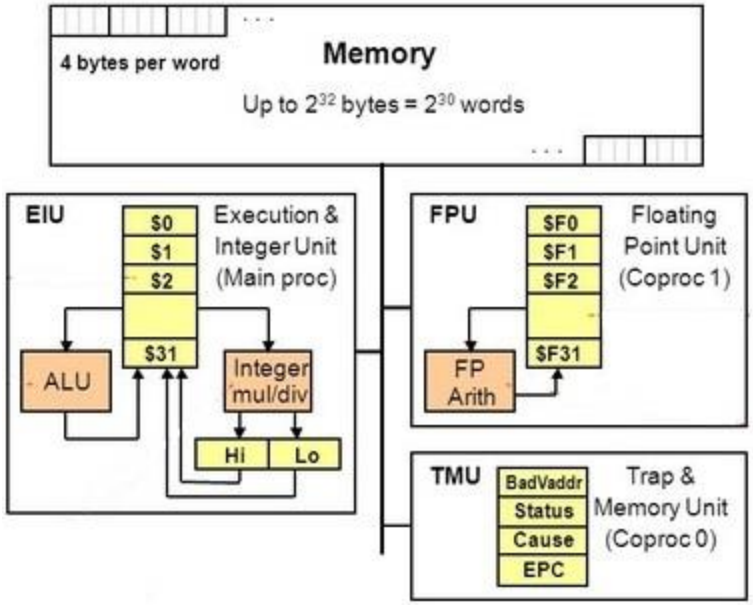
پردازنده میپس



Computer Architecture

Chapter Five

The MIPS Processor



Copyright Notice

Parts (text & figures of this lecture are adopted from:

- D. Patterson, J. Hennessy, “Computer Organization & Design, The Hardware/Software Interface, MIPS Edition”, 6th Ed., MK Publishing, 2020

Outlines

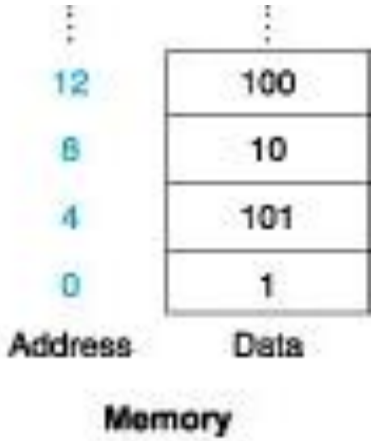
- General
- MIPS Registers
- MIPS Instructions
- MIPS Instruction Encoding
- MIPS Addressing Modes

MIPS-32 Processor

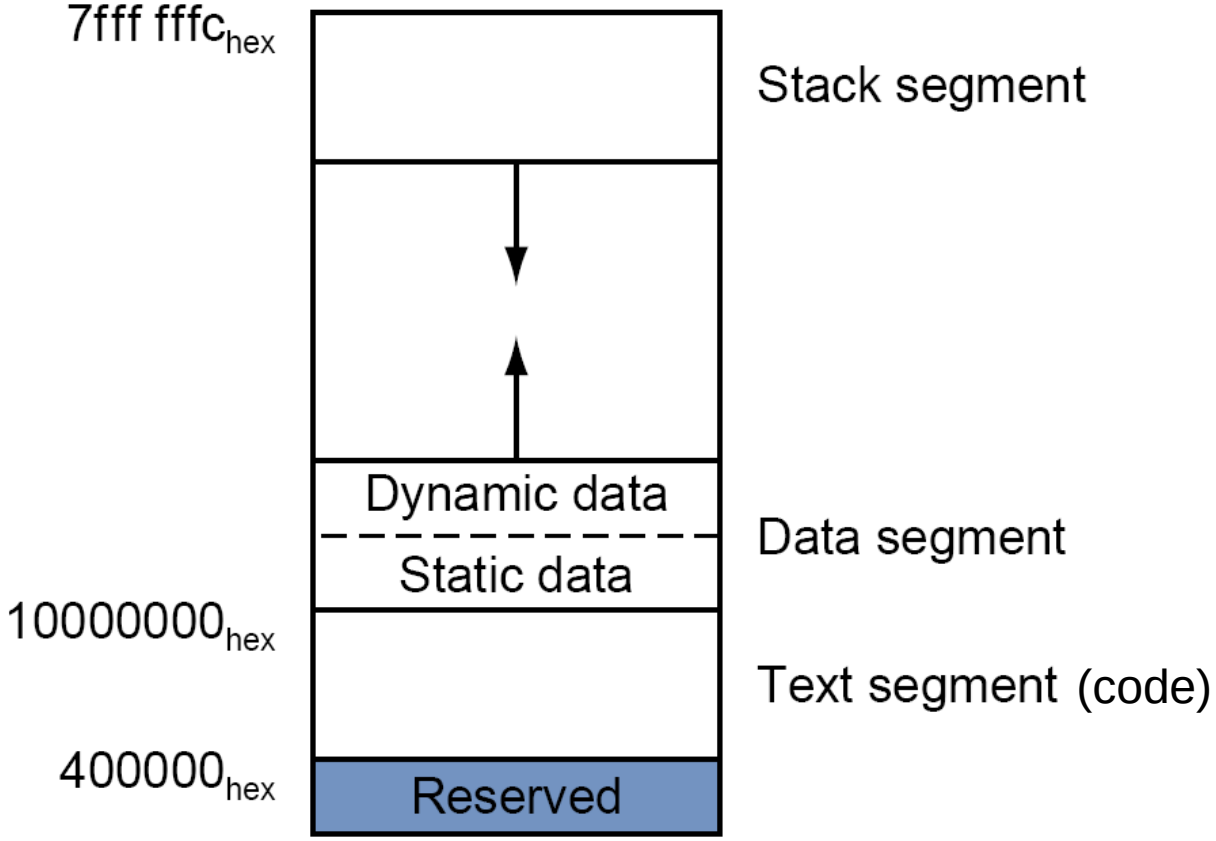
- 32-Bit Processor
  - Registers 32 bits
  - Arithmetic & logical operations 32 bits
- Load/Store ISA
  - Only load/store instructions can access memory
- 32-Bit Instruction Length

MIPS Memory Organization

- MIPS uses 4-bytes words
- Addresses are byte-based
- Words start at addresses that are multiples of 4
- This is called alignment restriction



Organization of MIPS Program



MIPS Design Principles

- Simplicity favors regularity
- Smaller is faster
- Make the common case fast
- Good design demands good compromises

Design Principle 1

- Simplicity favors regularity
- Regularity makes implementation simpler
- Simplicity enables higher performance at lower cost
- All arithmetic/logical instructions have three operands
  - two sources and one destination

```
add a,b,c    # a gets b+c
```

Design Principle 2

- Smaller is faster
- Arithmetic instructions use register operands
  - c.f. main memory with millions of locations
- MIPS has a 32 × 32-bit register file
  - Use for frequently accessed data
  - Numbered 0 to 31
  - 32-bit data called a “word”

Design Principle 3

- Make the common case fast
- Constant data specified in an instruction

```
addi $s1, $s0, 4
```

- Small constants are common
- Immediate operand avoids a load instruction

Design Principle 4

- Good design demands good compromises
- Different formats complicate decoding, but allow 32-bit instructions uniformly
- Keep formats as similar as possible

6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
op	rs	rt	rd	shamt	funct
op	rs	rt	16 bit address		
op	26 bit address				

MIPS Registers

- $s0-$s7
  - General (saved) registers
  - Preserved across a function call
- $t0-$t9
  - Temporary registers
  - Local to each subroutine
- $a0-$a3
  - Arguments for subroutine call
- $v0-$v1
  - Values for results of a subroutine call

MIPS Registers (cont.)

Register Number	Mnemonic Name	Conventional Use	Register Number	Mnemonic Name	Conventional Use
$0	$zero	Permanently 0	$24, $25	$t8, $t9	Temporary
$1	$at	Assembler Temporary (reserved)	$26, $27	$k0, $k1	Kernel (reserved for OS)
$2, $3	$v0, $v1	Value returned by a subroutine	$28	$gp	Global Pointer
$4–$7	$a0–$a3	Arguments to a subroutine	$29	$sp	Stack Pointer
$8–$15	$t0–$t7	Temporary (not preserved across a function call)	$30	$fp	Frame Pointer
$16–$23	$s0–$s7	Saved registers (preserved across a function call)	$31	$ra	Return Address

MIPS Instructions

- Arithmetic
- Logical & Shift
- Data Transfer
- Decision Making
  - Branch
  - Comparison
  - Subroutine Call

Arithmetic Instructions

- Addition / Subtraction
  - `add rd,rs,rt`
  - `sub rd,rs,rt`
  - `addi rd,rs,imm`
- Multiplication
  - `mult rs,rt`
  - `multu rs,rt`
- Division
  - `div rs,rt`
  - `divu rs,rt`

Logical Instructions

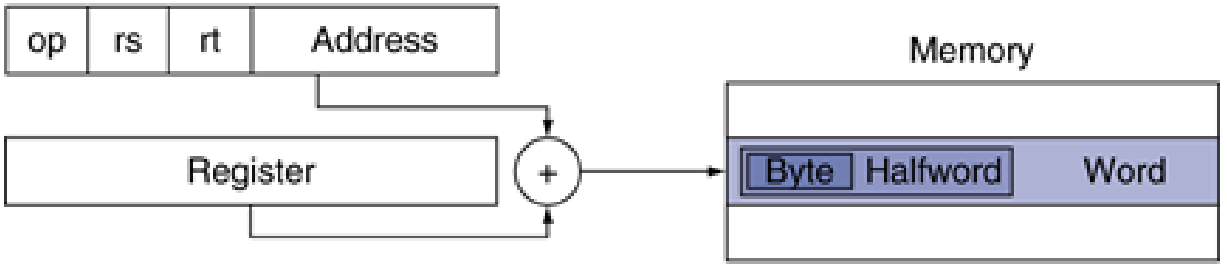
- AND
  - `and rd,rs,rt`
  - `andi rd,rs,imm`
- OR
  - `or rd,rs,rt`
  - `ori rd,rs,imm`
- NOR
  - `nor rd,rs,rt`

Shift Instructions

- Logical shift left/right (by specified number of bits)
  - `sll $rd,$rt,shamt`
  - `srl $rd,$rt,shamt`
- Logical shift left/right (by variable number of bits)
  - `sllv rd,rt,rs`
  - `srlv rd,rt,rs`
- Arithmetic Shift (keep sign while shifting)
  - `sra rd,rt,shamt`
  - `srav rd,rt,rs`

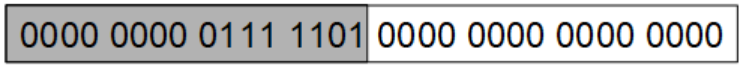
Data Transfer Instructions

- Load from memory (to registers)
  - `lw rt,cnst(rs)`
  - `lb rt,cnst(rs)/ lbu rt,cnst(rs)`
  - `lh rt,cnst(rs)/ lhu rt,cnst(rs)`



Data Transfer Instructions (cont.)

- Load from memory (to registers)
  - `lw rt,cnst(rs)`
  - `lb rt,cnst(rs)/ lbu rt,cnst(rs)`
  - `lh rt,cnst(rs)/ lhu rt,cnst(rs)`
- Store to memory (from registers)
  - `sw rt,cnst(rs)`
  - `sb rt,cnst(rs)`
  - `sh rt,cnst(rs)`
- Load upper immediate
  - `lui rs,imm`



Decision Making Instructions

- Branch
  - `beq rs,rt,L1`
  - `bne rs,rt,L1`

  } conditional

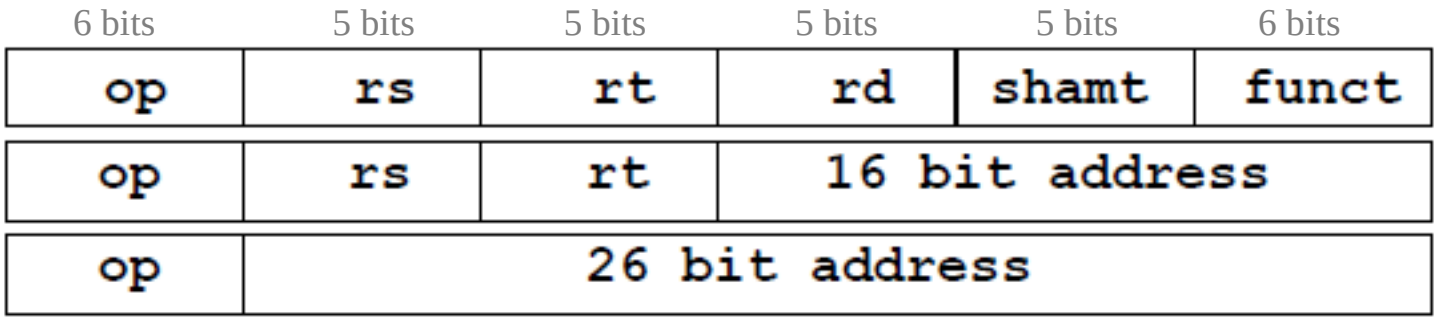
  - `j L1`
  - `jr reg`

  } unconditional

- Call Subroutine
  - `jal label`
- Comparison
  - `slt rd,rs,rt / sltu rd,rs,rt`
  - `slti rd,rs,imm / sltui rd,rs,imm`

MIPS Instruction Encoding

- MIPS instruction is exactly 32 bits
  - R-type (Register type)
  - I-type (Immediate type)
  - J-type (Jump type)



Example

Instruction	Format	Op	rs	rt	rd	shamt	funct
add	R	0	reg	reg	reg	0	32_{ten}
sub (subtract)	R	0	reg	reg	reg	0	34_{ten}
sll (logical shift left)	R	0	0	reg	reg	shamt	0
Instruction	**Format**	**Op**	**rs**	**rt**	**constant/address**		
add immediate	I	8_{ten}	reg	reg	constant		
lw (load word)	I	35_{ten}	reg	reg	address		
sw (store word)	I	43_{ten}	reg	reg	address		
Instruction	**Format**	**Op**	**address**				
j (jump)	J	2_{ten}	address				

op(31:26)								
28-26 31-29	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(000)	R-format	Bltz/gez	jump	jump & link	branch eq	branch ne	blez	bgtz
1(001)	add immediate	addiu	set less than imm.	set less than imm. unsigned	andi	ori	xori	load upper immediate
2(010)	TLB	FlPt						
3(011)								
4(100)	load byte	load half	lwl	load word	load byte unsigned	load half unsigned	lwr	
5(101)	store byte	store half	swl	store word			swr	
6(110)	load linked word	lwc1						
7(111)	store cond. word	swc1						

op(31:26)=000000 (R-format), funct(5:0)								
2-0 5-3	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(000)	shift left logical		shift right logical	sra	sllv		srlv	srav
1(001)	jump register	jalr			syscall	break		
2(010)	mfhi	mthi	mflo	mtlo				
3(011)	mult	multu	div	divu				
4(100)	add	addu	subtract	subu	and	or	xor	not or (nor)
5(101)			set l.t.	set l.t. unsigned				
6(110)								
7(111)								

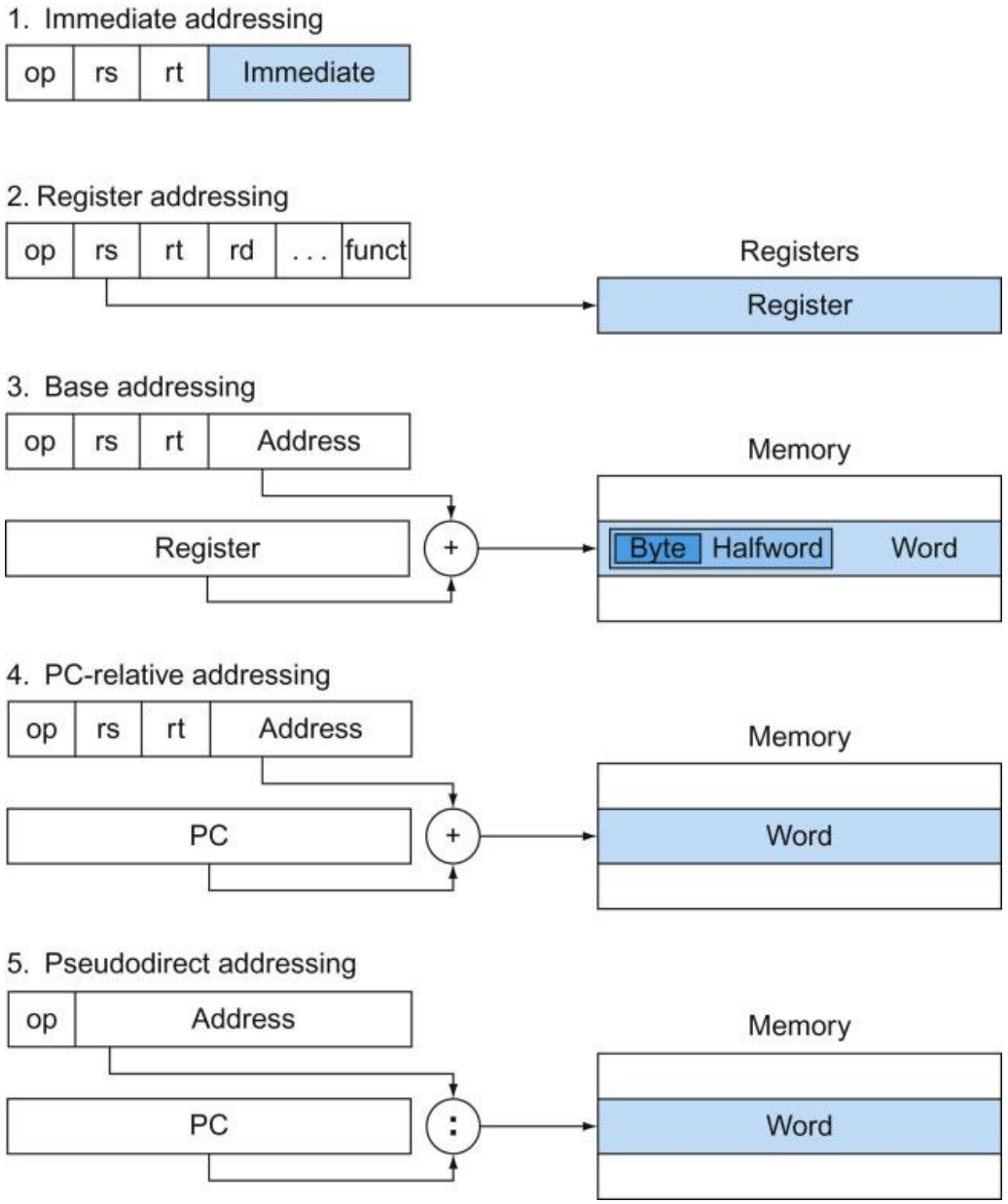
Example

Machine instruction: `00af8020hex`

op	rs	rt	rd	shamt	funct
000000	00101	01111	10000	00000	100000
R-type	$5	$15	$16	0	add

```
add $s0,$a1,$t7
```

Addressing Mode Summary



MIPS Processor Summary

- MIPS Registers
- MIPS Instructions
  - Arithmetic (add, sub, mul, div)
  - Data Transfer (load & store)
  - Logical & Shift
  - Decision Making
- MIPS Instruction Encoding
- MIPS Addressing Modes