Nível Máquina
Formatos de Instruções

IA32 e MIPS

Stored Program Concept

As instruções são números, armazenados em memória,
que são descodificados pela Unidade de Controlo do processador.

É possível descodificar estes números porque a Unidade de Controlo conhece o formato da informação que estes contêm.

Esta informação consiste geralmente nos seguintes campos:
- *opcode* – código que identifica a operação;
- identificação dos modos de endereçamento
- operandos (registos, valores imediatos)

Algumas arquitecturas incluem prefixos que modificam o comportamento da instrução.

Formato das Instruções : IA32

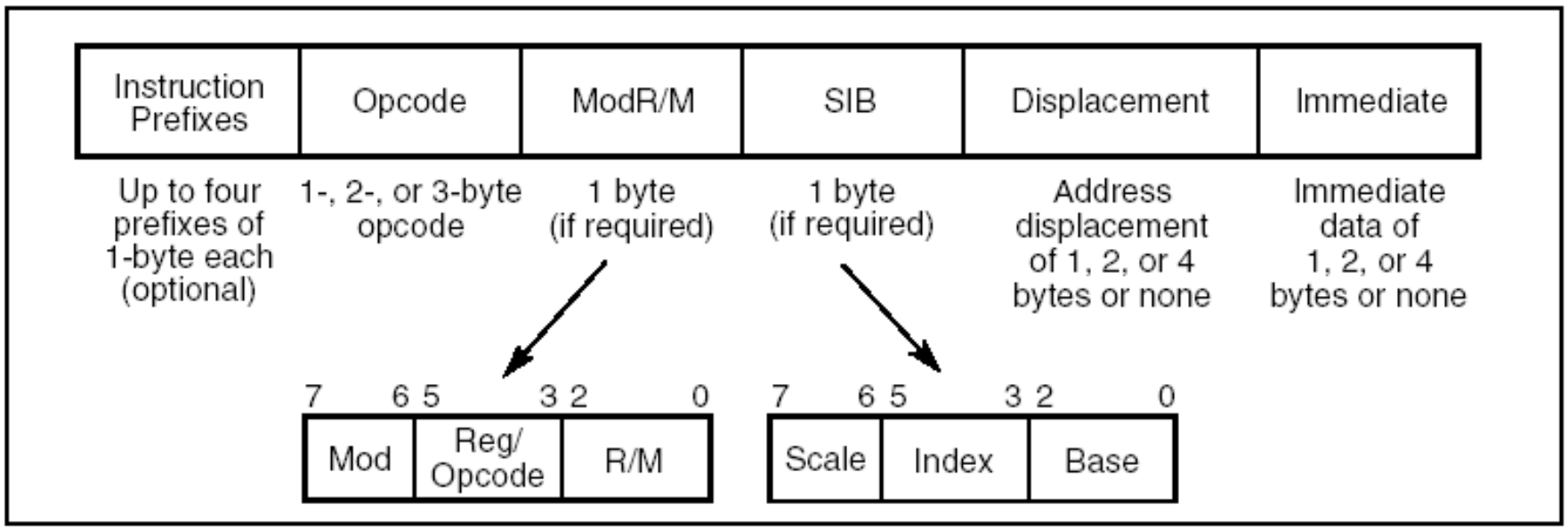


Figure 2-1. IA-32 Instruction Format

Instruções podem variar entre 1 a 17 bytes

Formatos das Instruções : IA32

ADDB $0x20, %AL

Opcode = 0x04 ⟶ 0x04 0x20

ADDL $0x5F0043, %EAX

Opcode = 0x05 ⟶ 0x05 0x43 0x00 0x5F 0x00

LOCK ADDL $0x5F0043, 0x08FFA021(%eax, %esi, 2)

Opcode = 0x81, Prefix = 0xF0 ⟶

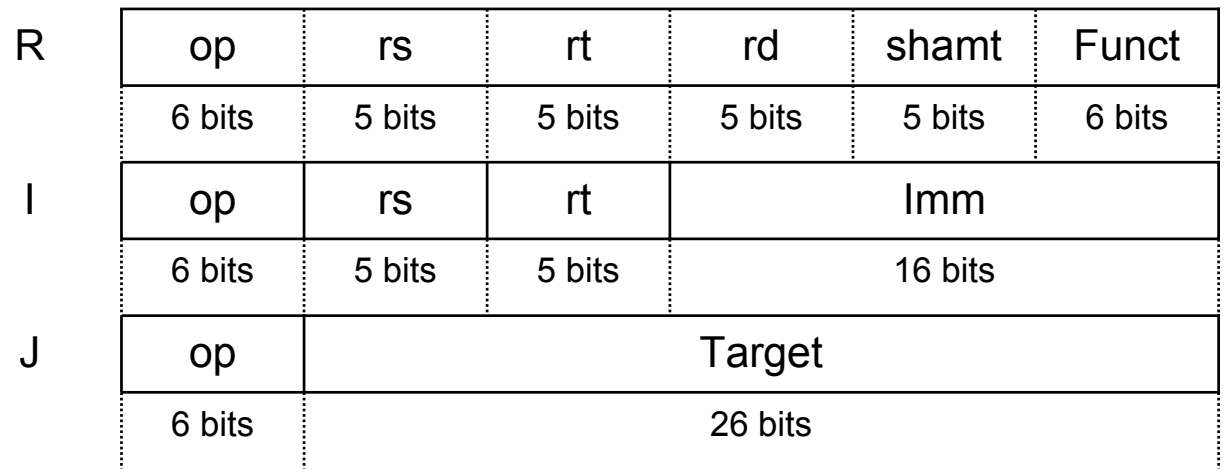
0xF0 0x81 0x84 0x70 0x21 0xA0 0xFF 0x08 0x43 0x00 0x5f 0x00

prefixo opcode ModR/M SIB deslocamento valor imediato

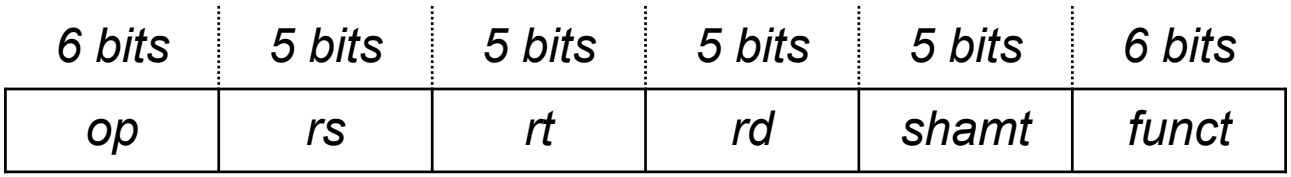
Formato de instruções – MIPS32

A simplicidade do formato das instruções permite o desenho de uma unidade de controlo mais simples, mais rápida e que facilita a incorporação de outras funcionalidades no processador.

Simplicidade: as instruções têm tamanho fixo e apenas 3 tipos diferentes



Tipo R



Op – código que identifica a instrução. Para R é sempre 0 (excepto rfe=0x10)

funct – identifica a função (operação)

rs, rt – números do dois registos que contêm os operandos

rd – número do registo que virá a conter o resultado

Número de shifts – apenas para sll, srl e sra (0 nas outras instruções)

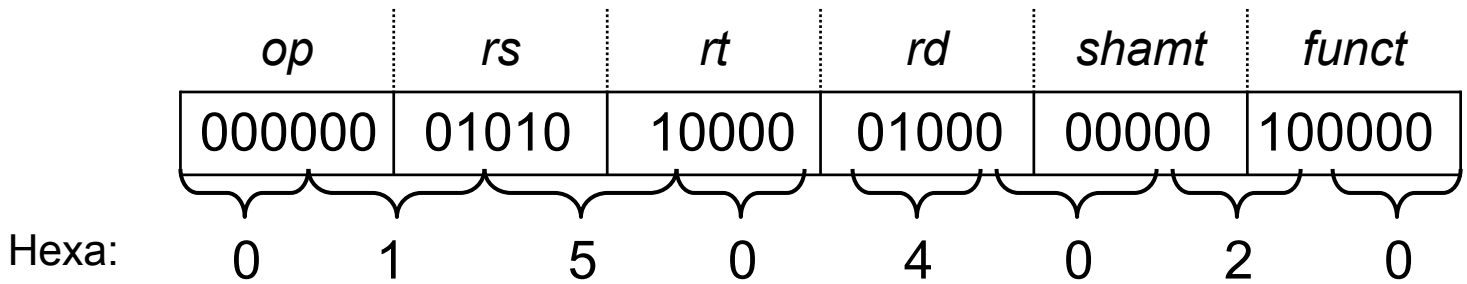
Tipo R

add $t0, $t2, $s0

op = 0x00 rt = $s0 = 16

funct = 0x20 rd = $t0 = 8

rs = $t2 = 10 shamt = 0

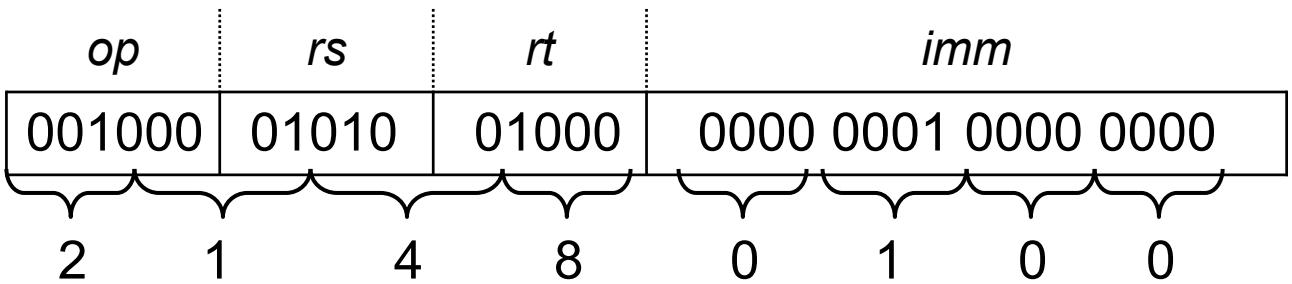


Tipo I

addi $t0, $t2, 256

op = 0x08 rt = $t0 = 8

rs = $t2 = 10 imm = 256

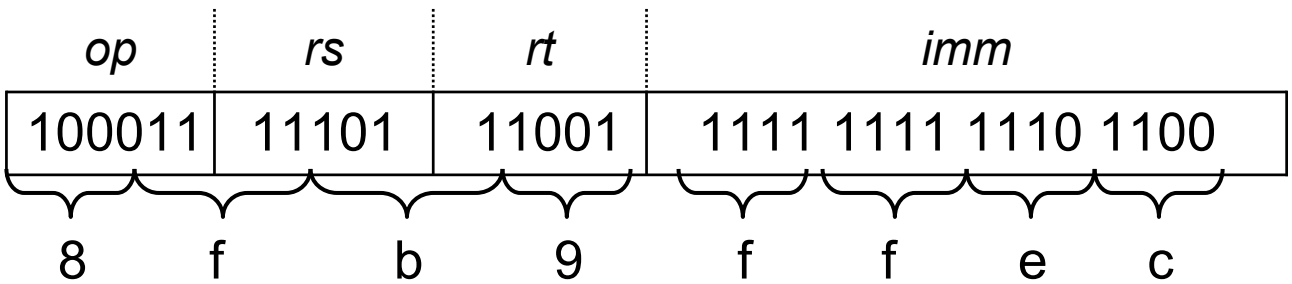


Tipo I

lw $t9, -20 ($sp)

op = 0x23 rt = $t9 = 25

rs = $sp = 29 imm = -20



Instruções de salto – *branch* e *jump*

Branch (Tipo I) – endereçamento relativo
A constante de 16 *bits* indica o número de instruções a saltar

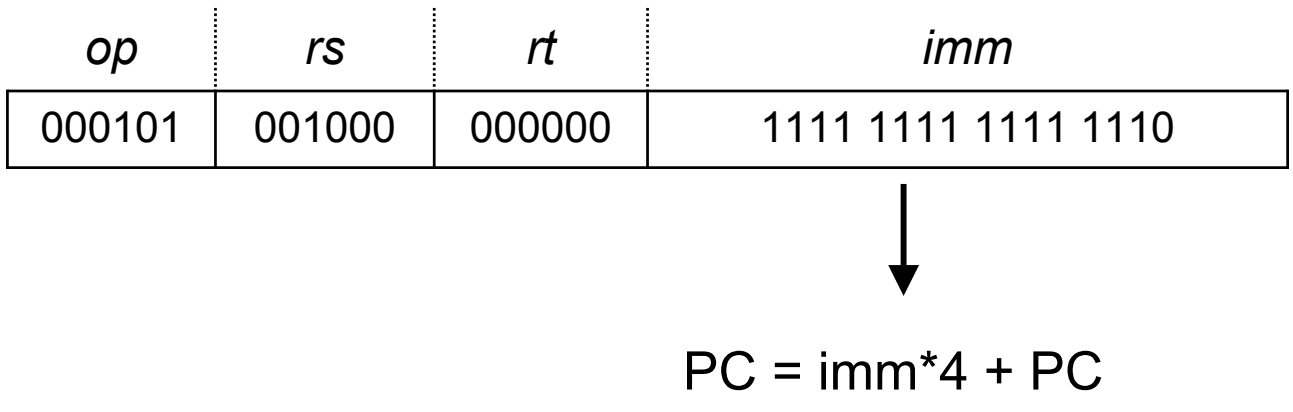
Jump (Tipo J) – endereçamento absoluto
A constante de 26 *bits* indica os 26 *bits* menos significativos do endereço de destino (excluindo os 2 bits menos significativos que são 00)

Jump register (Tipo R) – endereçamento absoluto
O registo indica o endereço de destino

Endereçamento relativo

```
for:
  subi $t0, $t0, -1
  bne $t0, $0, for
  addi $t0, $t0, -1
```

op = 05
rs = $t0 = 8
rt = $0 = 0



Endereçamento absoluto – Tipo J

j longe

Supondo que esta instrução está no endereço de memória 0XE0AA0000 e que longe é uma etiqueta que corresponde a 0xE0000008 então, depois do *fetch* e antes do *execute* teremos:

op	*target*
000010	0000 0000 0000 0000 0000 0000 10

PC = 1110 0000 1010 1010 0000 0000 0000 0100

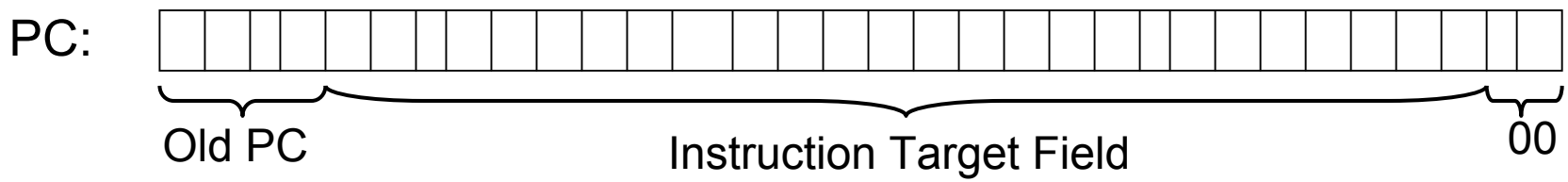
Após substituir os 26 *bits* apropriados do PC teremos

PC = 1110 0000 0000 0000 0000 0000 0000 1000

Endereçamento Absoluto –Tipo J

Endereço destino do jump é dado por:

- 4 bits mais significativos – são obtidos do PC
- 26 bits do meio – obtidos do campo target da instrução
- 2 bits menos significativos – 00 pois as instruções têm que estar em endereços múltiplos de 4



Para saltos que necessitem da especificação dos 32 bits do endereço de destino usar a instrução jr (jump register) carregando previamente o valor apropriado num registo

Valores Imediatos

Como lidar com instruções que tenham constantes de mais de 16 bits

```
addi $t0, $t0, 0xA00CD
```

1. Carregar a constante para `$at` usando `lui` e `ori`
2. Usar a versão R da instrução

```
lui $at, 0x000A
ori $at, $at, 0x00CD
add $t0, $t0, $at
```

$at: 0x000A00CD

Formato de Instruções

Tema	H & P	Manual Intel
MIPS - Instruções I e R	Sec 3.4, 3.5	
MIPS - Instruções J	Sec 3.8	
MIPS - Constantes > 16 bits	Sec 3.8	
Instruções Intel		Página da disciplina: “IA32 - Intel Architecture Software Developer's Manual. Volume 2: The Instruction Set Reference”