Universidade de Brasília
Departamento de Ciência da Computação

Aula 5
Assembly MIPS
linguagem de Máquina

Representação Numérica: Inteiros

- Binário sem sinal em N bits: $X = \sum_{i=0}^{N-1} b_i 2^i$

- Binário complemento de 2 em N bits
  - **Origem:** $X + (-X) = 2^N$
  - **Interpretação:** $X = -b_{N-1} 2^{N-1} + \sum_{i=0}^{N-2} b_i 2^i$
  - **Negação:** inverter e somar 1

Ex.: $5 = 0101$

$-5 = 1010 + 1 = 1011 = -2^3 + 2^1 + 2^0$

$X + \bar{X} = 111...111 = -1$

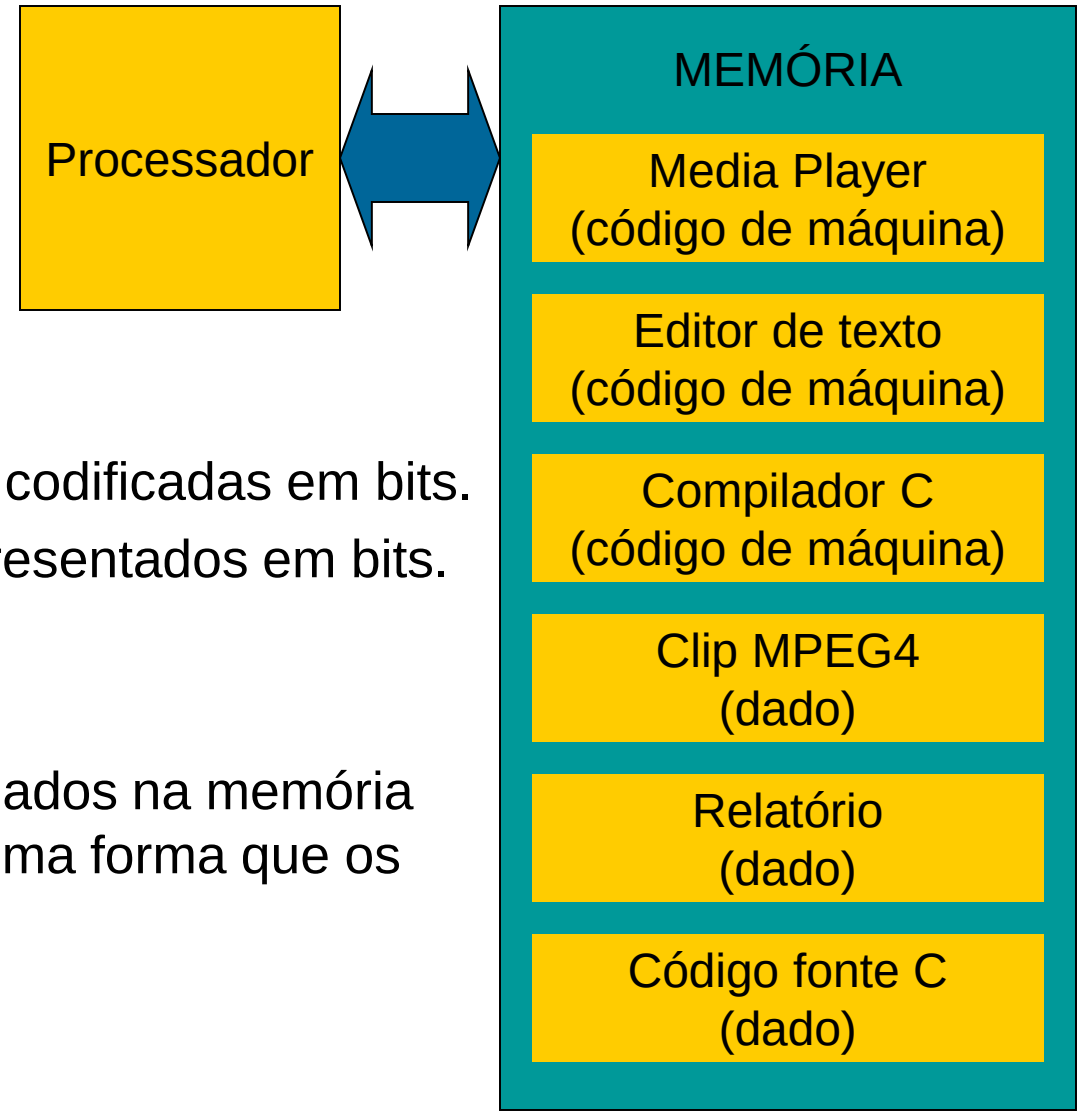
$-X = \bar{X} + 1$

  - **Extensão de Sinal :** repetir o MSB

Ex.: $5 = 0000\,0101$

$-5 = 1111\,1011$

Programa armazenado (conceito)



Todas as instruções são codificadas em bits.

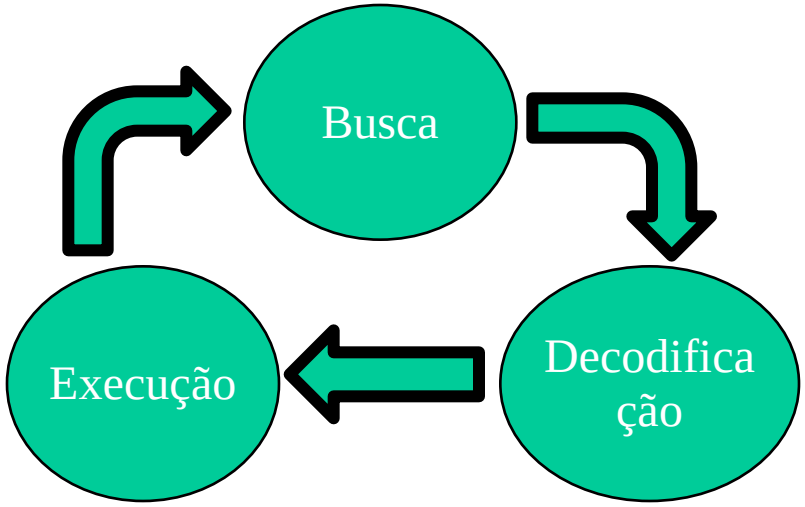
Todos os dados são representados em bits.

Programas são armazenados na memória para serem lidos da mesma forma que os dados.

Programa armazenado (conceito)

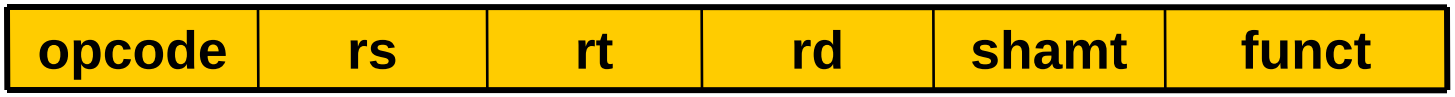
Ciclos de busca e execução:

- Instruções são buscadas na memória do endereço armazenado no registrador **PC : *Program Counter*** e colocadas no registrador **IR : *Instruction Register***
- Bits do registrador IR controlam as ações subsequentes necessárias à execução da instrução.
- Busca a próxima instrução e continua...



Linguagem de máquina

No MIPS, as instruções, assim como os registradores, também têm 32 bits de comprimento divididas em campos.



- *opcode* 6*bits* operação básica da instrução: *operation code*
- *rs* 5*bits* primeiro registrador de operando origem: *source*
- *rt* 5*bits* segundo registrador de operando origem: *target*
- *rd* 5*bits* registrador de operando destino: resultado: *destiny*
- *shamt* 5*bits* deslocamento: *shift amount*
- *funct* 6*bits* variação da operação: *function code*

Linguagem de máquina

Exemplo: **add $t0, $s1, $s2**

- Instrução **add**: opcode=0 funct=32 (vide guia de referência)
- registradores são identificados por seus números (vide tabelas): $t0=8, $s1=17, $s2=18

- **Formato Tipo-R de instrução:**

Campo	opcode	rs	rt	rd	shamt	funct
decimal	0	17	18	8	0	32
binário	000000	10001	10010	01000	00000	100000
Tamanho	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Outros exemplos de Tipo-R:

sub $t0,$t1,$t2 # subtração com sinal
addu $t0,$t1,$t2 # soma unsigned - sem sinal
sll $t0,$t1,2 # shift left logical

Linguagem de máquina

Formato de instrução para instruções com dados Imediatos.

Exemplo: **lw $t0, 32($s3) # Imediato positivo ou negativo**

- **Formato Tipo-I de instrução:**

op	rs	rt	Imm
35	19	8	32
100011	10011	01000	0000000000100000
6 bits	5 bits	5 bits	16 bits

Outros exemplos de Tipo-I:

lb $t0,Imm($s3) # store byte
lbu $t0,Imm($s3) # load byte unsigned
addi $t0,$t1,Imm # soma com imediato com sinal
addiu $t0,$t1,Imm # soma com imediato unsigned - sem sinal

Linguagem de Máquina

Controle de Fluxo: Desvio Incondicional

PC indica o endereço da próxima instrução a ser buscada na memória

j *Label* # Jump *Label:* PC=*Label*

jal *Label* # Jump and Link: $ra=PC+4; PC=*Label*

- **Formato Tipo-J de instrução:** **Ex.: j 1200**

op	Endereço
2	1200
000010	00000000000000010010110000
6 bits	26 bits

Outros exemplos instrução de desvio incondicional (Tipo-R)

jr $t0 # Jump Register: PC=[$t0]

jalr $t0 # Jump Register and Link: $ra=PC+4; PC=[$t0]

Linguagem de Máquina

- Controle de Fluxo: Desvio Condicional

  Instruções MIPS de desvio condicional são Tipo-I

  beq $t0, $t1, *Label* # Branch if Equal: $t0 == $t1 ? PC=*Label*

  bne $t0, $t1, *Label* # Branch if Not Equal: $t0 != $t1 ? PC=*Label*

Exemplo em C:

```
if (i!=j)
   h=i+j;
else
   h=i-j;
```

Assembly MIPS:

```
        bne $s4, $s5, Label1
        sub $s3, $s4, $s5
        j Label2
Label1: add $s3, $s4, $s5
Label2: ...
```

Em outras arquiteturas é comum o uso de *Flags* (Zero, Signal, Overflow, etc) para a realização de saltos condicionais.

Linguagem de Máquina

- Como implementar: <, >, <=, >= ?

Instrução MIPS: *Set on Less Than*

slt $t0,$t1,$t2 # $t0=1 se $t1<$t2; $t0=0 caso contrário (Tipo-R)
slti $t0,$t1,Imm # $t0=1 se $t1<Imm; $t0=0 caso contrário (Tipo-I)
sltu $t0,$t1,$t2 # comparação sem sinal (Tipo-R)
sltiu $t0,$t1,Imm # comparação imediato sem sinal (Tipo-I)

Apenas com estas instruções podemos montar várias estruturas de controle.
Ao montador é reservado o registrador $1 ($at) para essa tarefa

Ex.: Construa a pseudo-instrução *Branch If Less Than*
blt $t0,$t1,Label # se $t0 < $t1 então PC = Label

Uso de Constantes

- Constantes de até 16 bits: Uso das instruções tipo-I

Ex.:
```
addi $t0, $t1, 4        # $t0=$t1+4
slti $t0, $t1, 10       # $t0=($t1<10?1:0)
andi $t0, $t1, 6        # $t0=$t1 & 6
ori $t0, $t1, 4         # $t0=$t1 | 4
```

Pseudo-Instrução: *Load Immediate*

```
li $t0,0x1234           #  $t0=0x00001234
```

- Constantes de 16 até 32 bits: Uso de 2 instruções

Instrução: *Load Upper Imediate*

```
lui $t0, 0xF0CA         # $t0 = 0xF0CA<<16 = 0xF0CA0000
ori $t0,$t0,0xF0FA      # $t0 = $t0 | 0x0000F0FA
```

Pseudo-Instrução equivalente: *Load Address*

```
la $t0,0xF0CAF0FA       # $t0 = 0xF0CAF0FA
```

Endereços em desvios

***Label* : Endereço de 32 bits**

Instruções:

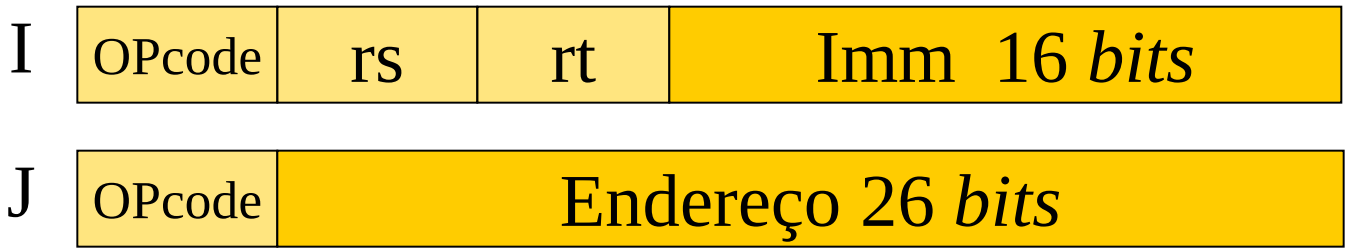
beq $t4, $t5, *Label* # Próxima instrução em *Label* se $t4 = $t5

bne $t4, $t5, *Label* # Próxima instrução em *Label* se $t4 ≠ $t5

j *Label* # Próxima instrução em *Label*

jal *Label* # $ra=PC+4; Próxima Instrução em *Label*

Formatos:



A representação dos endereços não têm 32 bits!!!

Endereços em desvios

- Instruções tipo-I, *beq* e *bne*, usam endereço relativo
  - maioria dos desvios condicionais são locais (Princípio da Localidade)
  - utiliza endereço relativo ao *Program Counter(PC)*

    PC = (*PC+4)+ExtSinal[Imm]<<2*

- Instruções tipo-J, *j e jal* , utilizam os 4 bits mais significativos do (PC+4) e concatenam ao *Endereço* deslocado 2 bits à esquerda.

  PC = { (PC+4)[31-28] , (*Endereço*<<2) }

  - limites blocos de 256 MiB (64Mi instruções).

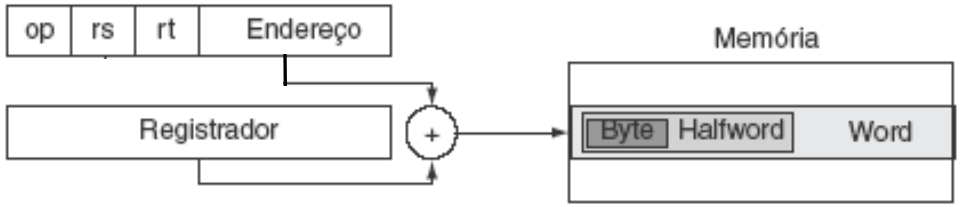
- O montador e o ligador precisam cuidar disso!

MIPS: Modos de endereçamento

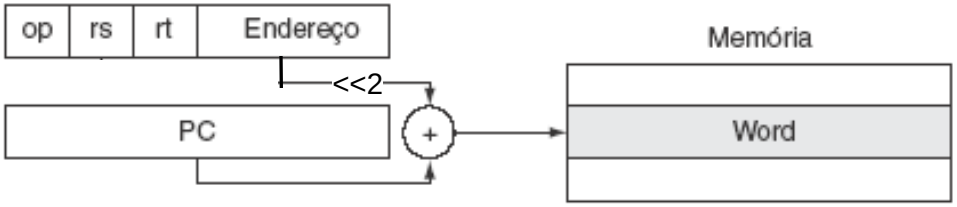
1. Endereçamento imediato

2. Endereçamento em registrador

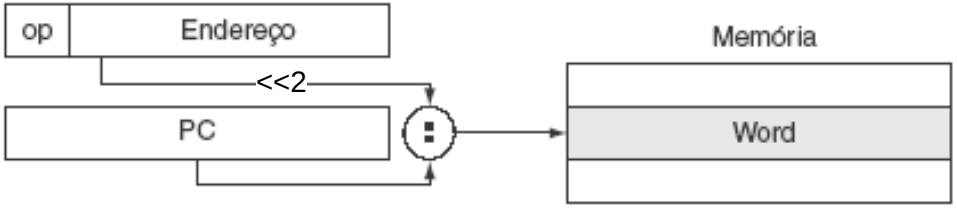
3. Endereçamento de base



4. Endereçamento relativo ao PC



5. Endereçamento pseudodireto



Ex.:
addi $t0,$t1,Imm
sra $t0,$t1,shamt

add $t0,$t1,$t2
jr $t0

lw $t0,Imm($t1)
lhu $t0,Imm($t1)
lbu $t0,Imm($t1)

beq $t0,$t1,Label

j Label
jal Label

Exemplo de Compilação

Linguagem C: while(save[i]==k) i++;

```
Loop:  sll $t1,$s3,2
       add $t1,$t1,$s6
       lw $t0,0($t1)
       bne $t0,$s5, Exit
       addi $s3,$s3,1
       j Loop
Exit:  …
```

0x00400000	0 000000	0 00000	19 10011	9 01001	2 00010	0 000000
0x00400004	0 000000	9 01001	22 10110	9 01001	0 00000	32 100000
0x00400008	35 100011	9 01001	8 01000	0 0000000000000000		
0x0040000C	5 000101	8 01000	21 10101	2 0000000000000010		
0x00400010	8 001000	19 10011	19 10011	1 0000000000000001		
0x00400014	2 000010	1048576 00000100000000000000000000				
0x00400018	...					

Na Memória:

```
0x00400000  00  13  48  80
0x00400004  01  36  48  20
0x00400008  8D  28  00  00
0x0040000C  15  15  00  02
0x00400010   22  73  00  01
0x00400014  08  10  00  00
0x00400018  …
```

O que aconteceria se no bne o imm=-1?