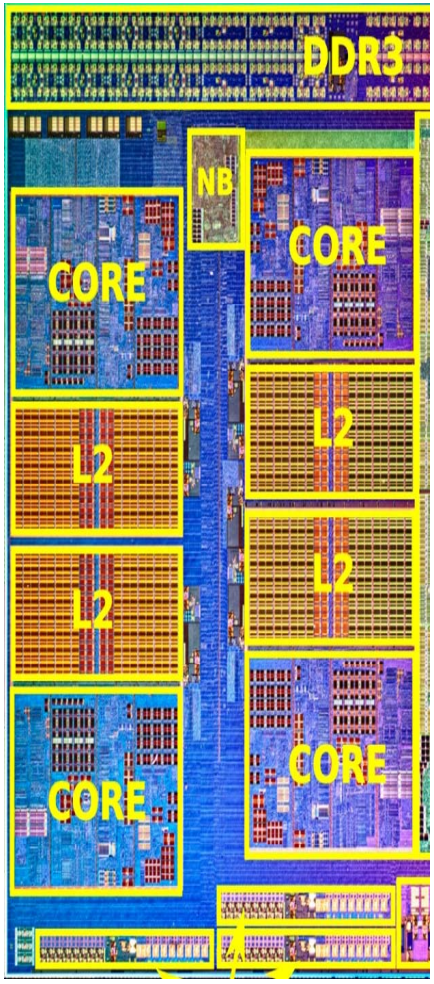
0273359 – Arquitetura e Organização de Computadores 1



Fonte: http://www.techspot.com/article/904-history-of-the-personal-computer-part-5/

Arquitetura MIPS

Luciano de Oliveira Neris
luciano@dc.ufscar.br

Adaptado de slides do prof. Marcio Merino Fernandes

Departamento de Computação
Universidade Federal de São Carlos

MIPS:Histórico

MIPS:Histórico

- MIPS, acrônimo para Microprocessor without Interlocked Pipeline Stages (Microprocessador sem estágios interligados de pipeline)

- Surgiu nos anos 80 a partir de um trabalho realizado por John Hennessy, na Universidade de Stanford.

MIPS:Histórico

- O trabalho tinha como objetivo explorar o padrão RISC, e é ainda considerado o mais elegante neste contexto
- O conceito introduzido por Hennessy foi tão bem sucedido que em 1984 foi formada a MIPS Technologies, Inc, a fim de comercializar os microprocessadores MIPS (R2000)
- Quase 100 milhões de processadores MIPS fabricados em 2009
- Atualmente esta presente em diversos produtos: Utilizada pela NEC, Nintendo, Cisco, Silicon Graphics, Sony, impressoras HP e Fuji, etc.

MIPS:Histórico

- MIPS é bastante utilizado para estudo da arquitetura de processadores por ser de notória simplicidade e clareza

- É a arquitetura base para construção de vários simuladores: R10k, ProcSIM, WinMIPS, MARS, WebSimple, SPIM, SimDE, entre outros.

- *Possui versões de 32 e 64 bits*

ISA MIPS

ISA MIPS

- A arquitetura MIPS é baseada em registrador, ou seja, a CPU utiliza apenas registradores para realizar as suas operações aritméticas e lógicas

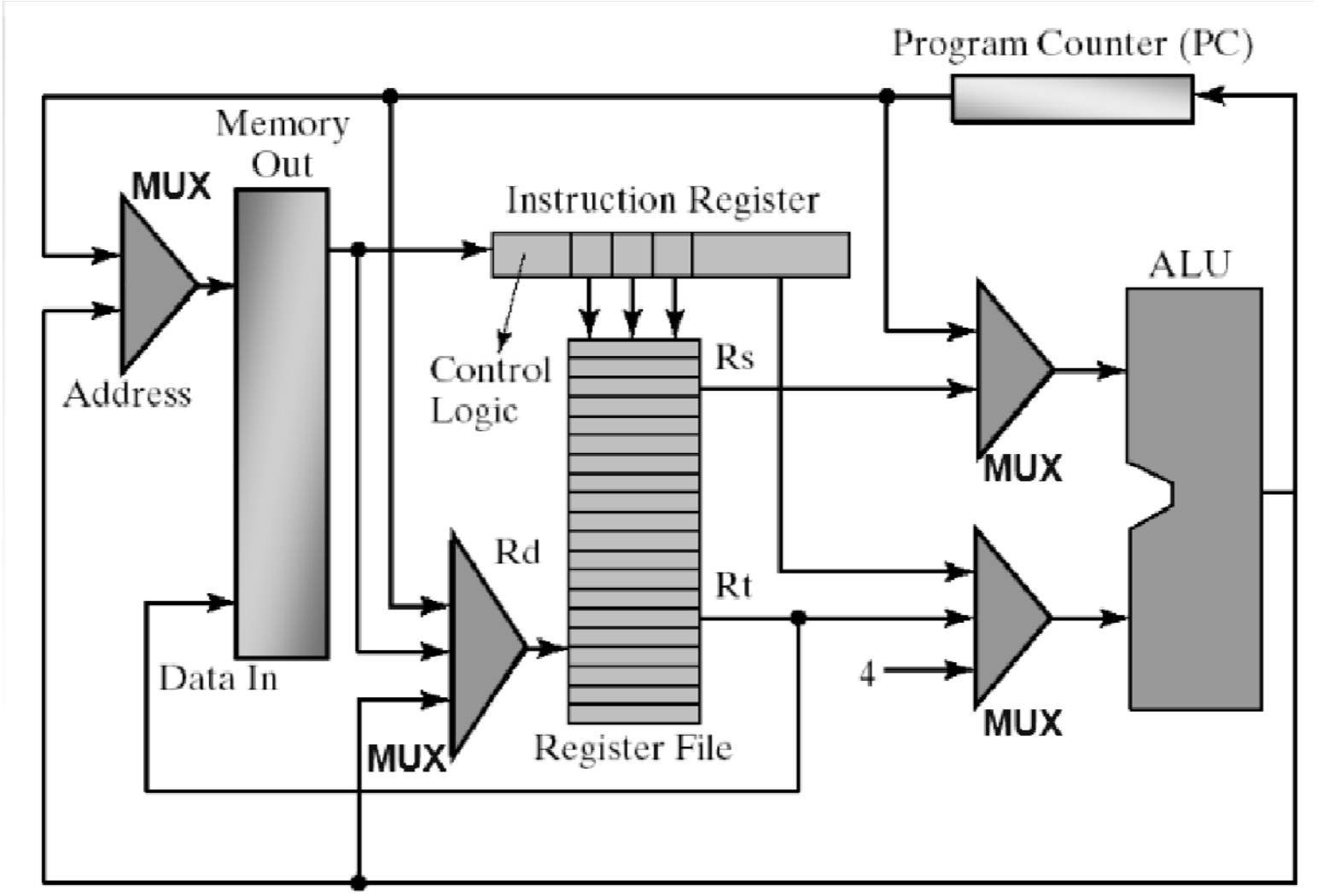
- É uma arquitetura do tipo load – store: as operações lógicas e aritméticas são executadas exclusivamente entre registradores da arquitetura ou entre constantes imediatas e registradores

- As operações de acesso à memória só executam ou uma leitura da memória (load) ou uma escrita na memória (store)

- O processador disponibiliza um conjunto relativamente grande de registradores para reduzir o número de acessos à memória externa

- As instruções possuem poucos formatos e são do mesmo tamanho

ISA MIPS



Fonte: Prof. Dr.-Ing. Frank Sill Torres http://www.cpdee.ufmg.br/~frank/

ISA MIPS

- Possui várias características/restrições
  - Todas as instruções possuem 32 bits: nenhuma instrução utiliza apenas dois ou três bytes de memória e nenhuma instrução pode ser maior do que 4 bytes
  - Todas as instruções possuem opcode de 6 bits
  - Todas as instruções aritméticas possuem 3 operandos
  - Utiliza a semântica big-endian para ordenar os bytes na memória

ISA MIPS: Registradores

- MIPS possui um banco de 32 registradores
  - Cada registrador comporta valores de 32 bits
  - Os registradores são identificados unicamente através do símbolo do cifrão ($) seguido de um identificador
  - Adota-se uma convenção que especifica quais registradores devem ser utilizados em certas circunstâncias
  - Os registradores PC (contador de programas) e IR (registrador de instrução) não fazem parte do banco de registradores

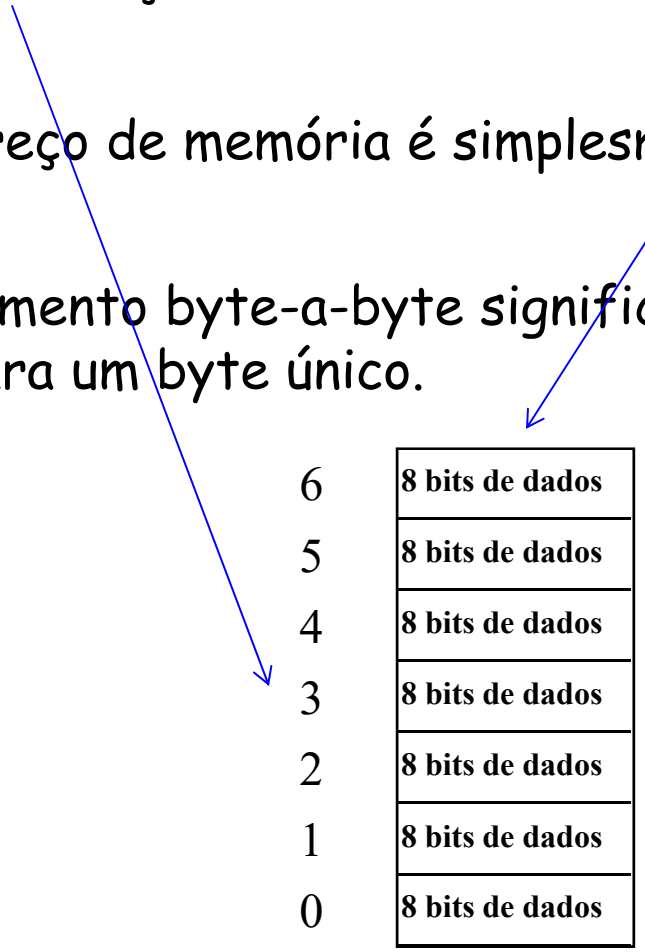
ISA MIPS: Registradores

#Registrador Físico	Nome Registrador no pgm MIPS	Convenção de uso (software)
$0	**$zero**	Constante = 0
$1	**$at**	Reservado p/ assembler
$2 - $3	**$v0 - $v1**	Retorna resultado de funções
$4 - $7	**$a0 - $a3**	Argumentos p/ funções
$8 - $15	**$t0 - $t7**	Temporários, (não preservados entre chamadas de funções)
$16 - $23	**$s0 - $s7**	Temporários, (preservados entre chamadas de funções)
$24 - $25	**$t8 - $t9**	Temporários, (não preservados entre chamadas de funções)
$26 - $27	**$k0 - $k1**	Reservado p/ OS kernel
$28	**$gp**	Global pointer
$29	**$sp**	Stack pointer
$30	**$fp**	Frame pointer
$31	**$ra**	Endereço de retorno de função (salvo na pilha pela instrução call)
$hi	**$hi**	Bits mais altos de resultado de 64 bits (remainder/div, high word/mult)
$lo	**$lo**	Bits mais baixos de resultado de 64 bits (quotient/div, low word/mult)

Organização da Memória

MIPS: Memória

- A memória pode ser visa como um grande vetor de bytes, cada um com um endereço hexadecimal.
- Um endereço de memória é simplesmente um índice desse vetor.
- Endereçamento byte-a-byte significa que cada índice do vetor aponta para um byte único.



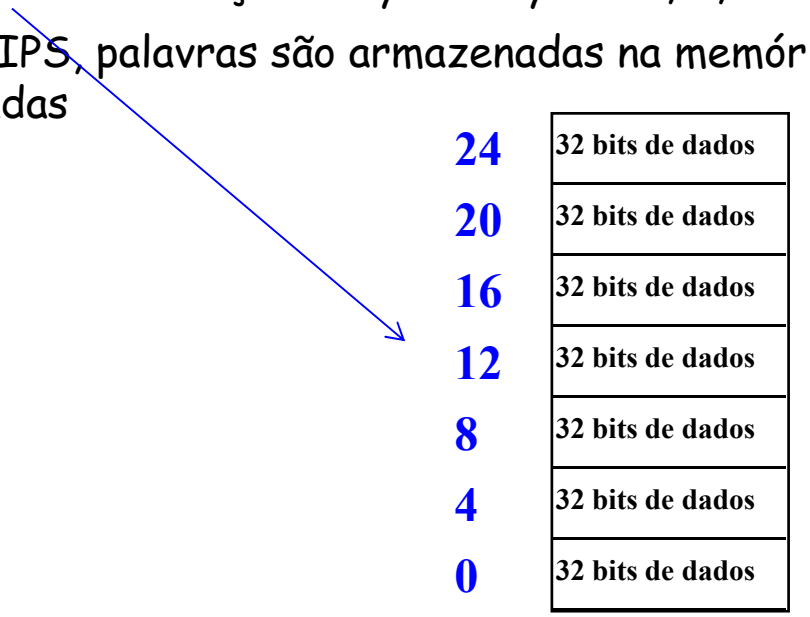
MIPS: Memória

- A maior parte dos dados utilizados em programas são maiores do que bytes.
- MIPS fornece as instruções lw/lh/lb and sw/sh/sb
  - w= word (32 bits), h= half-word (16 bits), b= byte (8 bits)

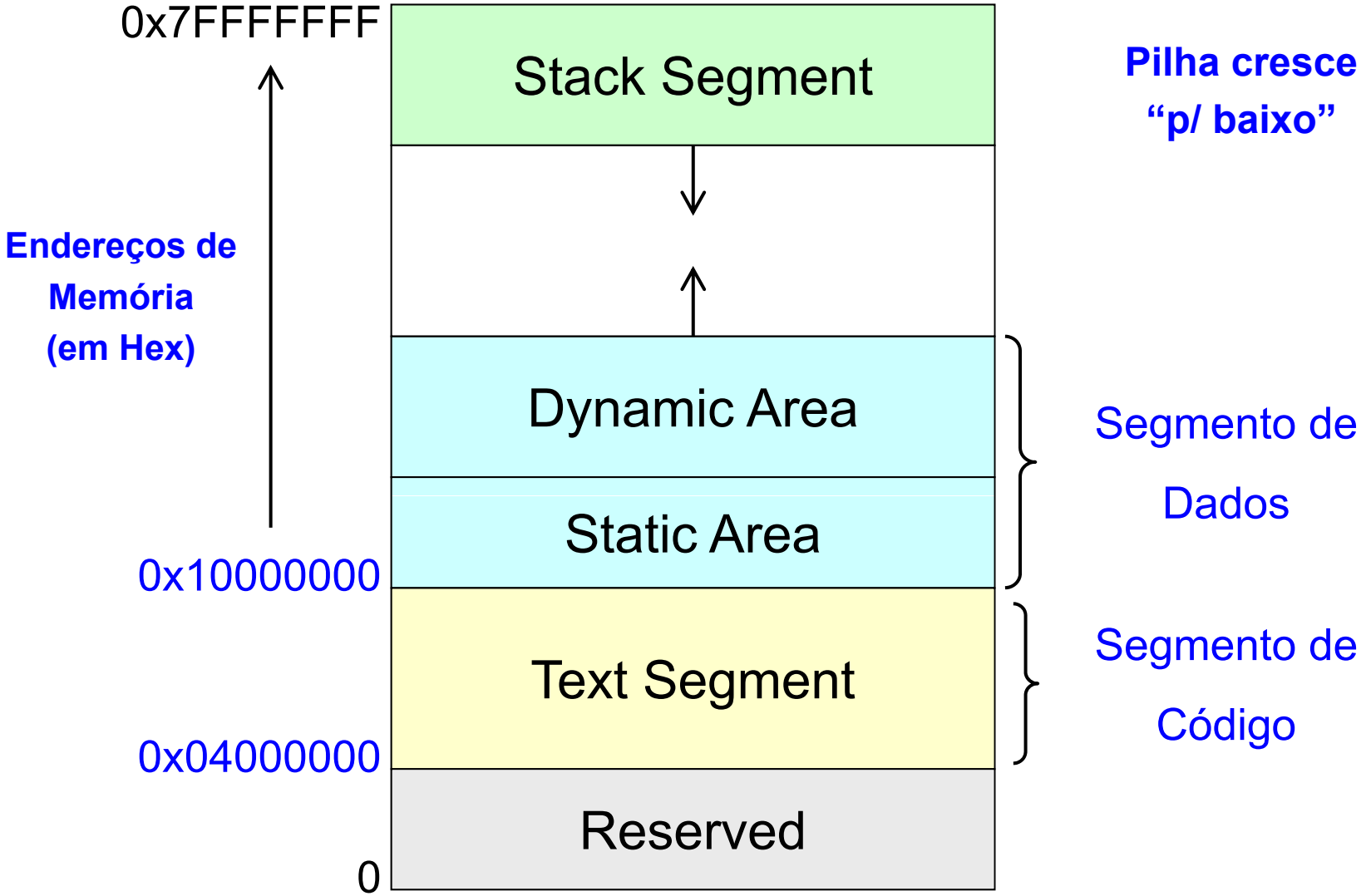
2^{32} bytes endereçados byte-a-byte = 0, 1, 2….2^{32}-1 endereços = 4GByte

2^{30} words endereçados byte-a-byte = 0, 4, 8 … 2^{30}-4 endereços= 1GWord

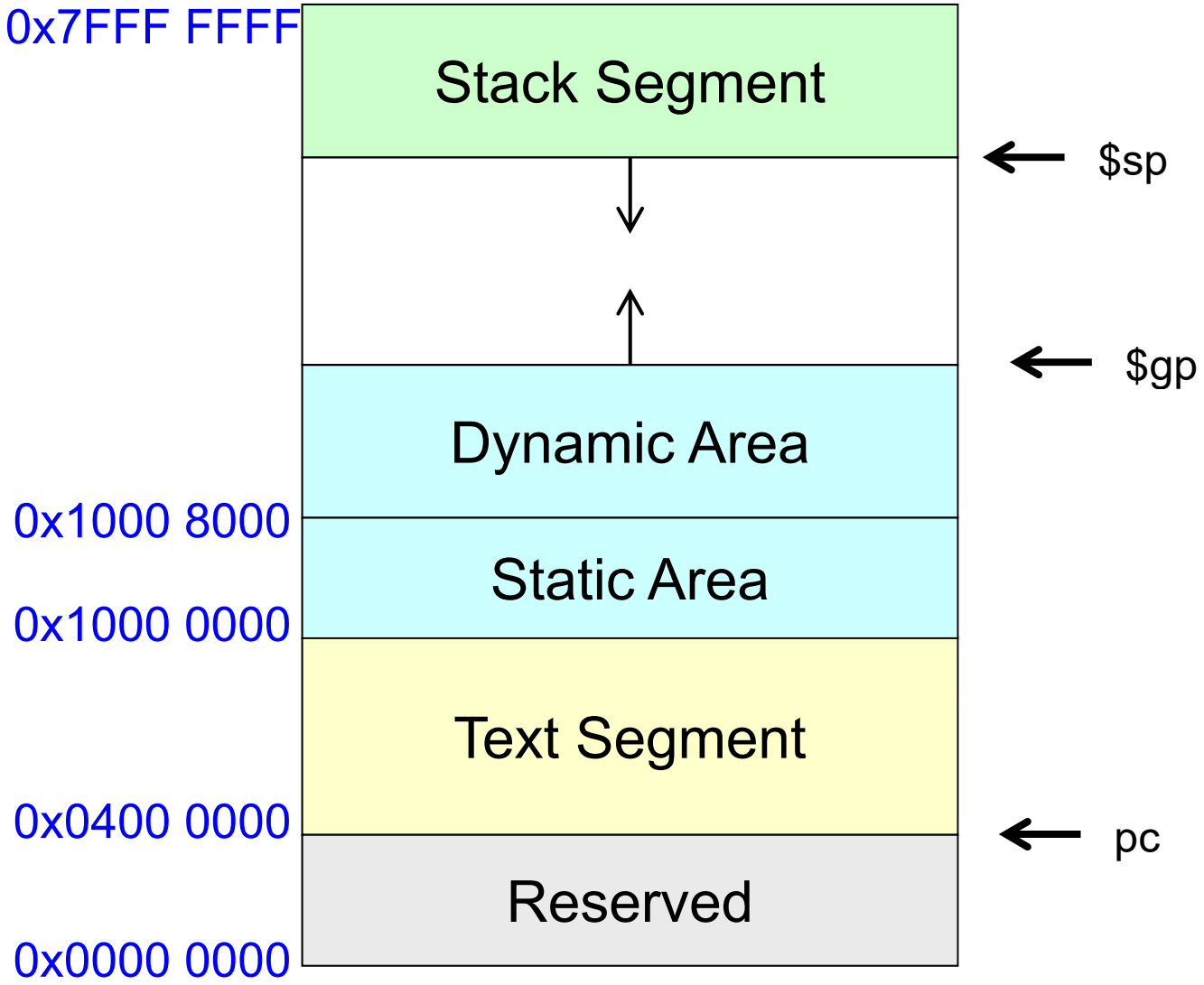
Em MIPS, palavras são armazenadas na memória em BIG-ENDIAN. Palavras são alinhadas



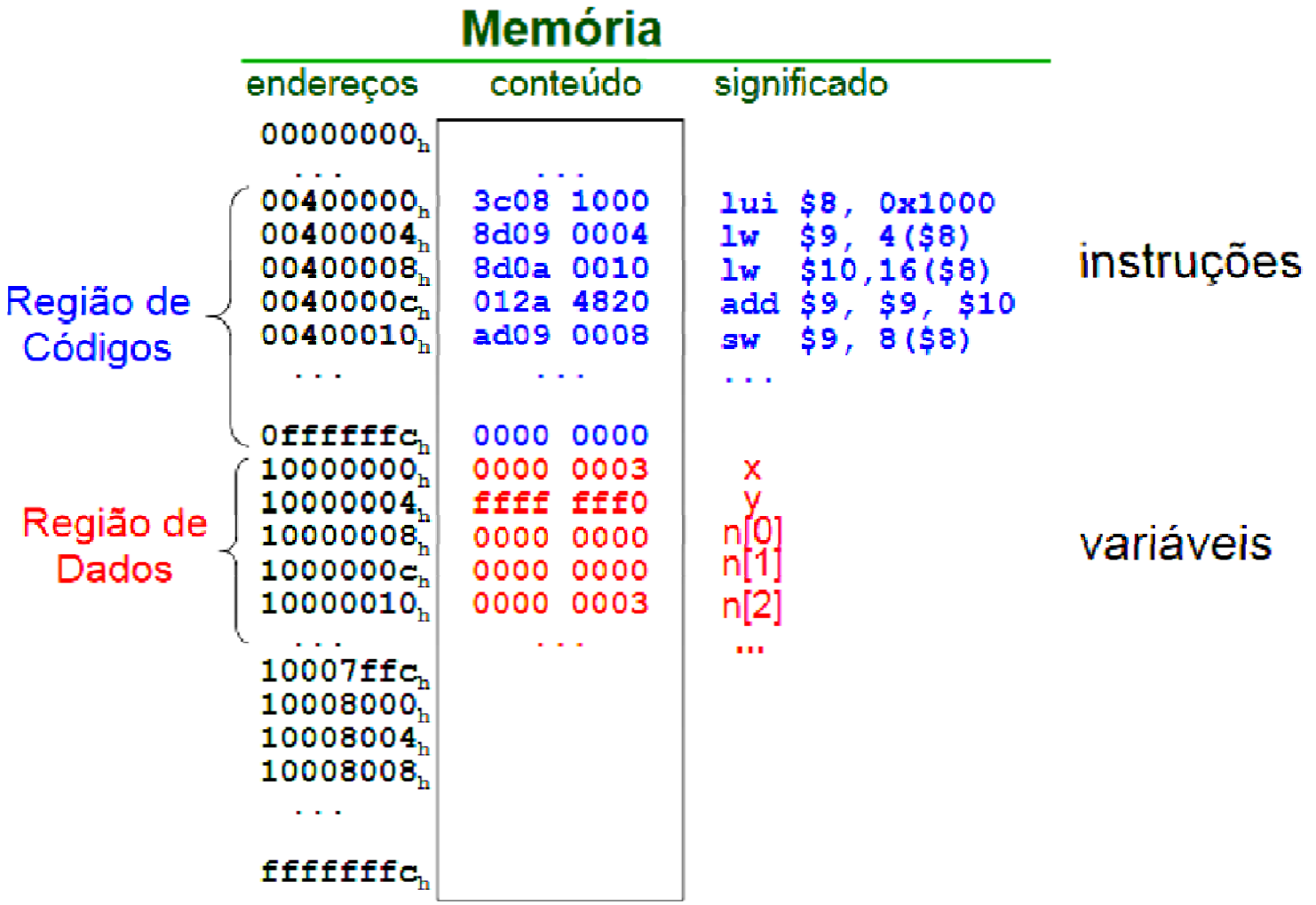
MIPS: Memória



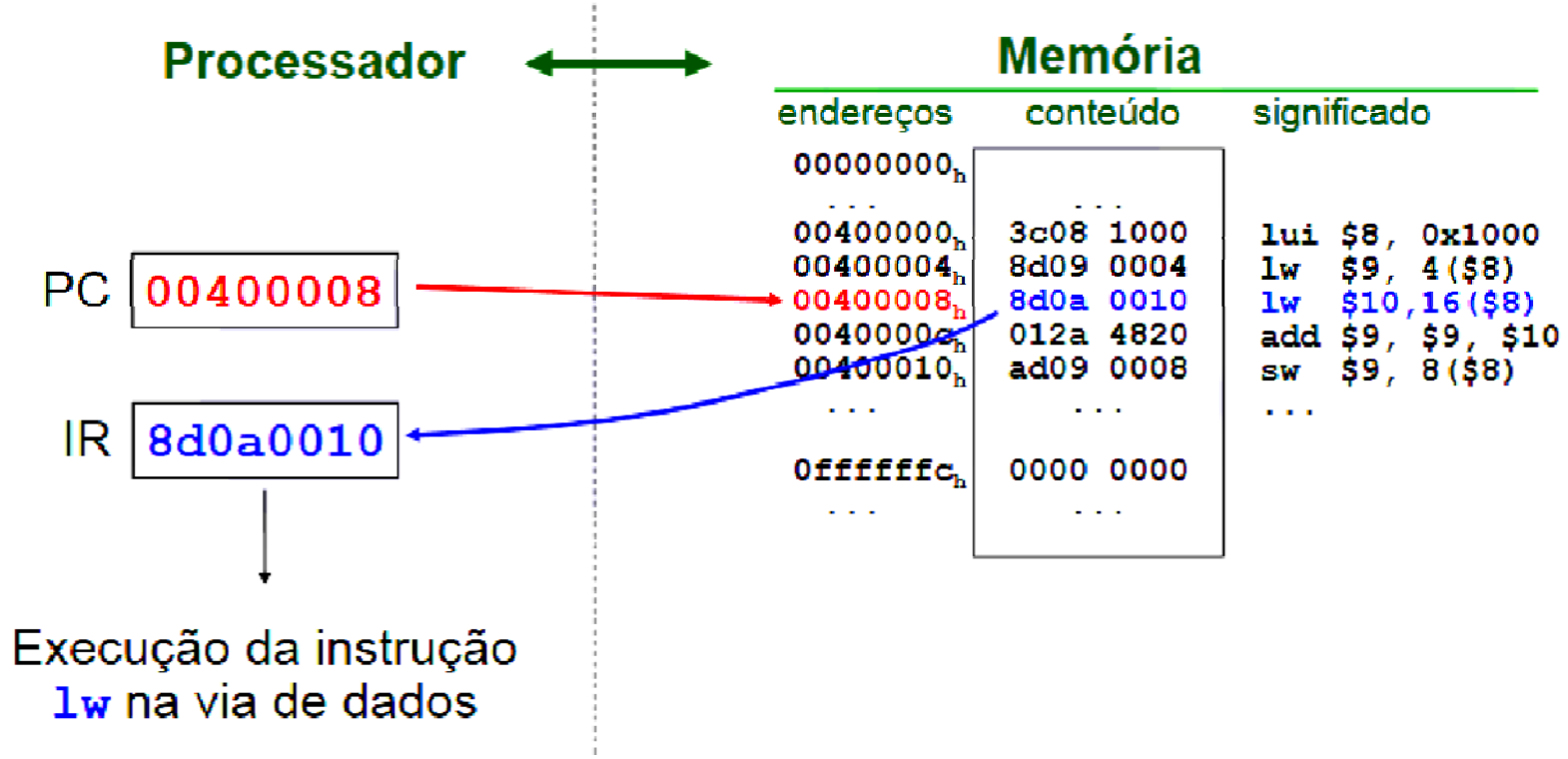
MIPS: Memória



MIPS: Memória



MIPS: Memória



Formato de Instruções

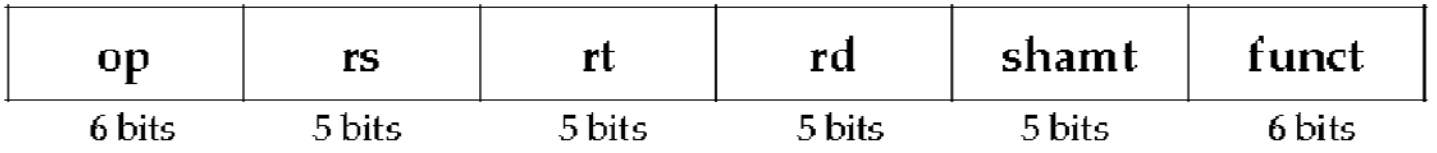
MIPS: Formato de Instruções

- Instruções
  - Todas as instruções possuem 32 bits
  - Todas as instruções possuem opcode de 6 bits
  - O modo de endereçamento é codificado no opcode

- As instruções podem ser classificadas em:
  - R-Type
  - I-Type
  - J-Type

MIPS: Formato de Instruções

- R-Type
  - Uso de registradores na instrução
  - Instruções registrador-registrador
  - As instruções do tipo R possuem opcode igual a 0.



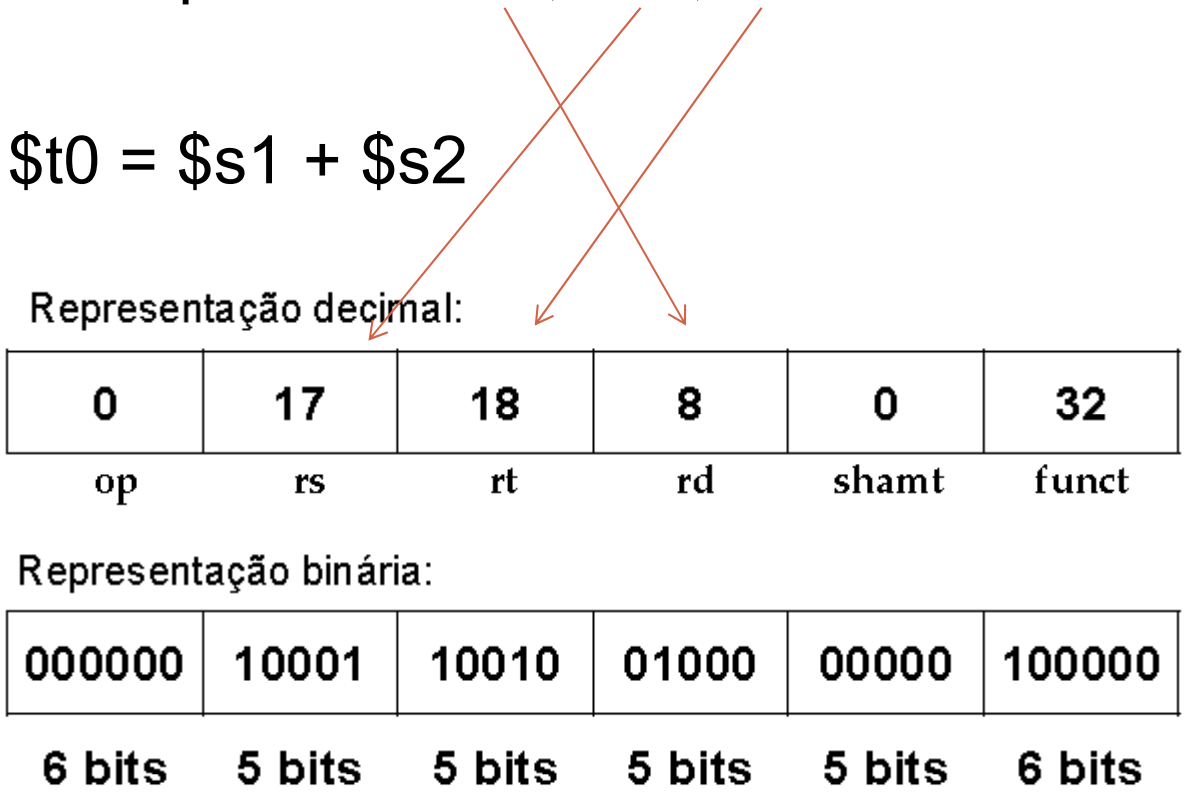
op – *opcode*: identifica a operação básica.
rs – primeiro operando (registrador).
rt – segundo operando (registrador).
rd – registrador de destino (resultado).
shamt – *shift amount*: define o tamanho do deslocamento.
funct – *function*: especifica a versão ou variante da operação indicada em op.

Formato R

MIPS: Formato de Instruções

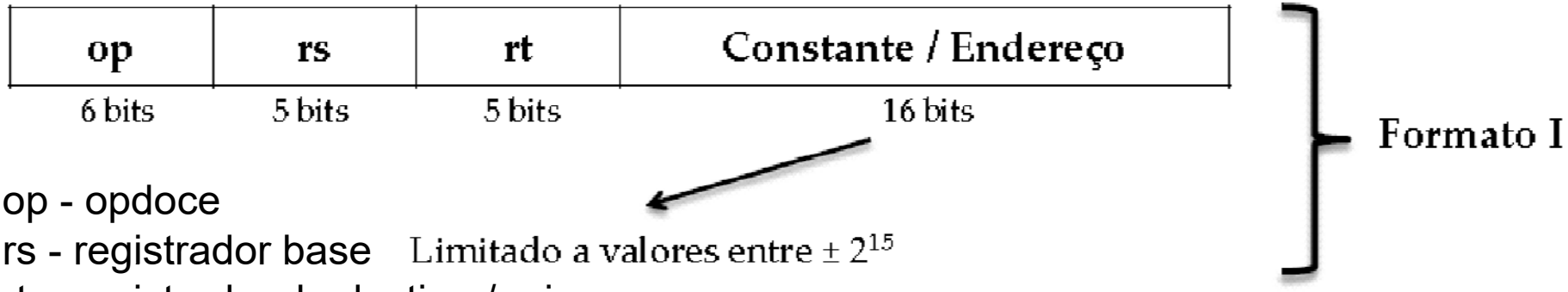
- R-Type
  - Exemplo: add $t0, $s1, $s2

$t0 = $s1 + $s2



MIPS: Formato de Instruções

- I-Type
  - Instruções envolvendo valor imediato
  - Um dos operandos pode ser representado na forma de um valor constante ou o endereço da palavra a ser acessada mantido dentro da própria instrução



op - opdoce
rs - registrador base
rt – registrador de destino / origem

Limitado a valores entre $\pm 2^{15}$

MIPS: Formato de Instruções

- I-Type:
  - Exemplo: Transferência de Dados

A[300] = h + A[300]

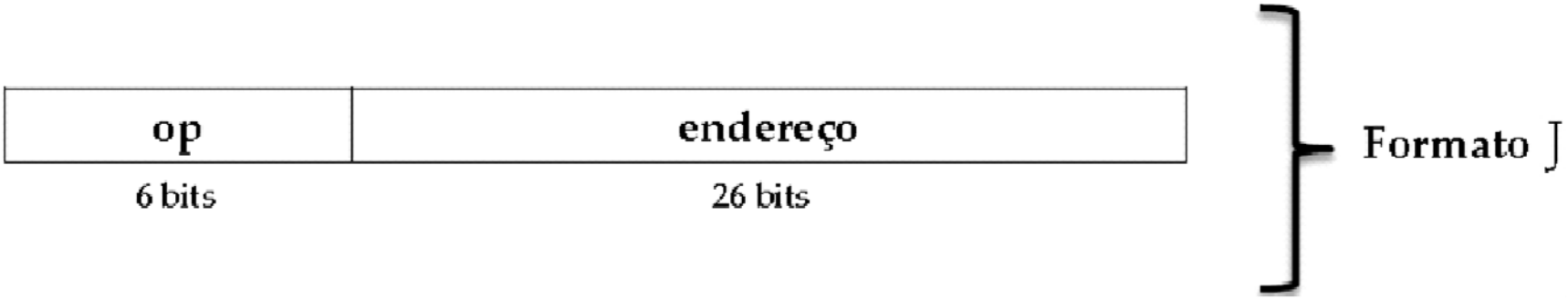
```
lw $t0, 1200 ($t1)  # $t0 = A[300]
add $t0, $s2, $t0   # $t0 = h + A[300]
sw $t0, 1200 ($t1) # A[300]  = $t0
```

```
op  rs rt  endereço
35  9  8   1200

op  rs  rt  rd  shmat  func
0   18  8  8   0        32

op  rs  rt  endereço
43  9   8   1200
```

100011	01001	01000	0000 0100 1011 0000
000000	10010	01000	01000 | 00000 | 100000
101011	01001	01000	0000 0100 1011 0000

MIPS: Formato de Instruções

- J-Type
  - Instruções de desvio incondicional (jump)
  - O desvio sempre é realizado



op - opcoce
endereço (target) - local da memória a saltar (onde esta a próxima instrução a ser executada).

MIPS: Formato de Instruções

- J-Type
  - Exemplo

```
        add $s0, $s1, $s2
        j Exit
        sub $s0, $s1, $s2
Exit:
```

MIPS: Formato de Instruções

- Machine Codes

Mnemonic	Meaning	Type	Opcode	Funct
`add`	Add	R	0x00	0x20
`addu`	Add Unsigned	R	0x00	0x21
`and`	Bitwise AND	R	0x00	0x24
`div`	Divide	R	0x00	0x1A
`divu`	Unsigned Divide	R	0x00	0x1B
`jr`	Jump to Address in Register	R	0x00	0x08
`mfhi`	Move from HI Register	R	0x00	0x10
`mflo`	Move from LO Register	R	0x00	0x12
`mfc0`	Move from Coprocessor 0	R	0x10	NA
`mult`	Multiply	R	0x00	0x18
`multu`	Unsigned Multiply	R	0x00	0x19
`nor`	Bitwise NOR (NOT-OR)	R	0x00	0x27
`xor`	Bitwise XOR (Exclusive-OR)	R	0x00	0x26
`or`	Bitwise OR	R	0x00	0x25
`slt`	Set to 1 if Less Than	R	0x00	0x2A
`sltu`	Set to 1 if Less Than Unsigned	R	0x00	0x2B
`sll`	Logical Shift Left	R	0x00	0x00
`srl`	Logical Shift Right (0-extended)	R	0x00	0x02

Mnemonic	Meaning	Type	Opcode	Funct
`sra`	Arithmetic Shift Right (sign-extended)	R	0x00	0x03
`sub`	Subtract	R	0x00	0x22
`subu`	Unsigned Subtract	R	0x00	0x23
`j`	Jump to Address	J	0x02	NA
`jal`	Jump and Link	J	0x03	NA
`addi`	Add Immediate	I	0x08	NA
`addiu`	Add Unsigned Immediate	I	0x09	NA
`andi`	Bitwise AND Immediate	I	0x0C	NA
`beq`	Branch if Equal	I	0x04	NA
`bne`	Branch if Not Equal	I	0x05	NA
`lbu`	Load Byte Unsigned	I	0x24	NA
`lhu`	Load Halfword Unsigned	I	0x25	NA
`lui`	Load Upper Immediate	I	0x0F	NA
`lw`	Load Word	I	0x23	NA
`ori`	Bitwise OR Immediate	I	0x0D	NA
`sb`	Store Byte	I	0x28	NA
`sh`	Store Halfword	I	0x29	NA
`slti`	Set to 1 if Less Than Immediate	I	0x0A	NA
`sltiu`	Set to 1 if Less Than Unsigned Immediate	I	0x0B	NA
`sw`	Store Word	I	0x2B	NA

https://en.wikibooks.org/wiki/MIPS_Assembly/Instruction_Formats