GBC046–Arq. e Org. de Computadores II

ISA – Instruction Set Architecture (Revisão)

Universidade Federal de Uberlândia
Faculdade de Computação
Prof. Dr. rer. nat. Daniel D. Abdala

Na Aula Anterior ...

- Anatomia de um Sistema Computacional;
- Placa Mãe;
- North Bridge e South Bridge;
- SIO;
- Barramentos:
  - PCI;
  - PCI-E;
- Inicialização do Sistema:
  - BIOS;
  - POST;
  - Carregamento do SO.

2

Nesta Aula

- Conceitos Gerais;
- Tipos de Dados;
- Formatos de Instruções;
- Instruções suportadas;
- Registradores;
- Modos de Endereçamento;
- Arquitetura de Memória;
- Interrupções, Traps e Exceções.

3

Conceitos

- ISA – Instruction Set Architecture
  - Idealização de como o µprocessador deve funcionar;
  - Visão funcional do µprocessador;
  - µprocessador do ponto de vista do programador;
- ISA passa a impressão de tratar apenas do conjunto de instruções, mas na verdade é mais que isso:
  - Tipos de Dados;
  - Formatos de Instruções;
  - Instruções suportadas;
  - Registradores;
  - Modos de Endereçamento;
  - Arquitetura de Memória;
  - Interrupções, Traps e Exceções;
  - Modos de Operação.

4

Tipos de Dados

- µProcessadores são máquinas lógicas e aritméticas;
- Atuam diretamente sobre representações numéricas;
- Circuitos elétricos para processar números reais são essencialmente distintos daqueles que processam números naturais;
- Espaço representacional limitado 16, 32, 64 bits;

5

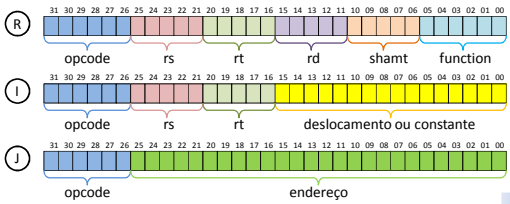
Tipos de Dados

- Na ISA do MIPS3000 (MIPS1)
  - Naturais ($\mathbb{N}$) – Inteiros não sinalizados
    - Double Word – 64 bits (resultados de mults)
    - Word – 32 bits
    - Half – 16 bits
    - Byte – 08 bits (e.g. usado para caracteres ASCII)
  - Inteiros ($\mathbb{Z}$) – Inteiros sinalizados
    - 16 bits, complemento de dois
    - 32 bits, complemento de dois
  - Reais ($\mathbb{R}$) – ponto flutuante (IEEE754)
    - 32 bits – registradores especiais
    - 64 bits – registradores especiais
- Como outros conjuntos reais são suportados?
  - Racionais ($\mathbb{Q}$), Complexos ($\mathbb{C}$), etc...

6

Formatos de Instruções

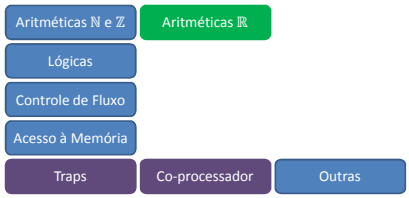
- **Formato de Instrução →** Como codificar instruções em uma sequência de bits;
- **No MIPS1 →** apenas 3 formas possíveis (tipos de formatos);



7

Conjunto de Instruções

- Operações que o µprocessador é capaz de processar;
- Subdivididas para melhor entendimento:



8

Instruções Aritméticas (ℕ e ℤ)

OP	Descrição	Exemplo
add	Adição com trap no overflow (não suportado pelo C)	add $rd, $rs, $rt
addi	Adição com trap no overflow – segundo operando cte.	addi $rt, $rs, cte
addiu	Adição sem trap no overflow – segundo operando cte.	addi $rt, $rs, cte
addu	Adição sem trap no overflow	addu $rd, $rs, $rt
aui	Adiciona a cte. Aos bits mais significativos	aui $rt, $rs, cte
clo	Conta # de zeros até achar um 1	clo $rd, $rs
clz	Conta # de uns até achar um 0	clz $rd, $rs
div	Divisão com overflow (LO ← quociente | HI ← resto)	div $rs, $rt
divu	Divisão Natural sem overflow	divu $rs, $rt
madd	Multiplica e adiciona	madd $rs, $rt
maddu	Multiplica e adiciona (Naturais)	maddu $rs, $rt

9

Instruções Aritméticas (ℕ e ℤ)

OP	Descrição	Exemplo
mfhi	Move o dado contido no registrador HI	mfhi $rs
mflo	Move o dado contido no registrador LO	mflo $rs
msub	Multiplica e subtrai	msub $rs, $rt
msubu	Multiplica e subtrai (Naturais)	msubu $rs, $rt
mtlo	Move o dado contido em $rs para o registrador LO	mtlo $rs
mthi	Move o dado contido em r$ para o registrador HI	mthi $rs
mul	Multiplicação s/ overflow	mul $rd, $rs, $rt
mult	Multiplicação (hi ← $32bits_{MSB}$ | lo ← $32bits_{LSB}$)	mult $rs, $rt
multu	Multiplicação Natural (hi ← $32bits_{MSB}$ | lo ← $32bits_{LSB}$)	multu $rs, $rt
sub	Subtração com overflow	sub $rd, $rs, $rt
subu	Subtração sem overflow	sub u $rd, $rs, $rt

10

Instruções Aritméticas (ℝ)

OP	Descrição	Exemplo
abs.d	$f2 ← \|$f4| valor absoluto (precisão dupla)	abs.d $f2, $f4
abs.s	$f0 ← \|$f1| valor absoluto (precisão simples)	abs.d $f0, $f1
add.d	$f2 ← $f4 + $f6 Soma em precisão dupla	add.d $f2, $f4, $f6
add.s	$f0 ← $f1 + $f2 Soma em precisão simples	add.d $f0, $f1, $f2
c.eq.d	$f2 == $f4 ? CF0 ← 1 : CF0 ← 0	c.eq.d $f2, $f4
c.eq.s	$f0 == $f1 ? CF0 ← 1 : CF0 ← 0	c.eq.s $f0, $f1
c.le.d	$f2 <= $f4 ? CF0 ← 1 : CF0 ← 0	c.le.d $f2, $f4
c.le.s	$f0 <= $f1 ? CF0 ← 1 : CF0 ← 0	c.le.s $f0, $f1
c.lt.d	$f2 < $f4 ? CF0 ← 1 : CF0 ← 0	c.lt.d $f2, $f4
c.lt.s	$f0 < $f1 ? CF0 ← 1 : CF0 ← 0	c.lt.s $f0, $f1
ceil.w.d	Aproxima para cima para o inteiro (word)	ceil.w.d $f1, $f2

11

Instruções Aritméticas (ℝ)

OP	Descrição	Exemplo
ceil.w.s	Aproxima para cima para o inteiro (word)	ceil.w.s $f0, $f1
cvt.d.s	Converte de precisão simples para dupla	cvt.d.s $f2, $f1
cvt.d.w	Converte de inteiro 32 bits para precisão dupla	cvt.d.w $f2, $f1
cvt.s.d	Converte de precisão dupla para simples	cvt.s.d $f1, $f2
cvt.s.w	Converte inteiro 32bits para single	cvt.s.w $f0, $f1
cvt.w.d	Converte de precisão dupla para inteiro 32 bits	cvt.w.d $f1, $f2
cvt.w.s	Converte de precisão simples para inteiro 32 bits	cvt.w.s $f0, $f1
div.d	Divisão em ponto flutuante precisão dupla	div.d $f2, $f4, $f6
div.s	Divisão em ponto flutuante precisão simples	div.s $f0, $f1, $f3
floor.w.d	Arredonda para o inteiro 32 bits para baixo (double)	floor.w.d $f1, $f2
floor.w.s	Arredonda para o inteiro 32 bits para baixo (single)	floor.w.s $f0, $f1

12

Instruções Aritméticas (ℝ)

OP	Descrição	Exemplo
mov.d	Move o double em f4-f3 para f2-f3	mov.d $f2, $f4
mov.s	Move o single em f1 para f0	mov.s $f0, $f1
movf	Move t2 para t1 se CF0 = 0	movf $t1, $t2
movt	Move t2 para t1 se CF0 = 1	movt $t1,$t2
mul.d	Multiplicação em precisão dupla	mul.d $f2,$f4,$f6
mul.s	Multiplicação em precisão simples	mul.s $f0,$f1,$f3
neg.d	Negação double	neg.d $f2,$f4
neg.s	Negação single	neg.s $f0,$f1
sub.d	Subtração double	sub.d $f2,$f4,$f6
sub.s	Subtração single	sub.s $f0,$f1,$f3

13

Instruções Lógicas

OP	Descrição	Exemplo
and	E bit a bit	and $rd, $rs, $rt
andi	E bit a bit – segundo operando cte	andi $rt, $rs, cte
nor	NÃO OU bit a bit	nor $rd, $rs, $rt
or	OU bit a bit	or $rd, $rs, $rt
ori	OU bit a bit – segundo operando cte	ori $rt, $rs, cte
sll	Deslocamento para a esquerda (multiplicação base 2)	sll $rt, $rs, [0:31]
slr	Deslocamento para a direita (divisão base 2)	slr $rt, $rs, [0:31]
sllv	Deslocamento para a esquerda (multiplicação base 2)	sllv $rd, $rs, $rt
slrv	Deslocamento para a direita (divisão base 2)	slrv $rd, $rs, $rt
slt	Seta se menor que (sinalizado)	slt $rd, $rs, $rt

14

Instruções Lógicas

OP	Descrição	Exemplo
slti	Seta se menor que imediato (sinalizado)	slti $rt, $rs, cte
sltiu	Seta se menor que (não sinalizado)	sltiu $rt, $rs, cte
sltu	Seta se menor que imediato (não sinalizado)	sltu $rd, $rs, $rt
xor	OU EXCLUSIVO bit a bit	xor $rd, $rs, $rt
xori	OU EXCLUSIVO bit a bit – segundo operando cte.	xori $rt, $rs, cte

15

Instruções de Controle de Fluxo

OP	Descrição	Exemplo
beq	(=) Salta se igual	beq $rs, $rt, offset
bgez	(≥0)Salta se maior ou igual a zero	bgez $rs, $rt, offset
bgezal	(≥0)Salta se maior ou igual a zero (usado em subrotinas) ($ra ←PC+4)	bgezal $rs, $rt, offset
bgtz	(>0)Salta se maior que zero	bgtz $rs, $rt, offset
blez	(≤0)Salta se menor ou igual a zero	blez $rs, $rt, offset
bltz	(<0) Salta se menor que zero	bltz $rs, $rt, offset
bltzal	(<0) Salta se menor que zero ($ra ←PC+4)	bltzal $rs, $rt, offset
bne	(≠) salta se diferente	bne $rs, $rt, offset
j	Salto incondicional	j offset
jal	Salto incondicional e liga ($ra ←PC+4)	jal offset
jalr	Salto incondicional (registrador) Salva em $ra o endereço da instrução de retorno	jalr $rs
jr	Salta para endereço em registrador	jr $rs
movn	rd ← rs se rt == 0	movn $t1, $t2, $t3

16

Instruções de Acesso a Memória

OP	Descrição	Exemplo
lb	Carrega byte da memória (rt ←MEM[rs+offset])	lb $rt, offset($rs)
lbu	Carrega byte da memória (rt ←MEM[rs+offset])	lbu $rt, offset($rs)
lh	Carrega half word da memória (rt ←MEM[rs+offset])	lh $rt, offset($rs)
lhu	Carrega half word da memória (rt ←MEM[rs+offset])	lhu $rt, offset($rs)
lui	Carrega a cte nos 16 bits mais significativos do registrador	Lui $rt, cte
lw	Carrega word da memória (rt ←MEM[rs+offset])	lw $rt, offset($rs)
sb	Salva byte na memória (MEM[rs+offset] ← rt)	sb $rt, offset($rs)
sh	Salva half word na memória (MEM[rs+offset] ← rt)	sh $rt, offset($rs)
sw	Salva wordna memória (MEM[rs+offset] ← rt)	sw $rt, offset($rs)
mtc1	Move uma word de um reg. de propósito geral para o C1	mtc1 $f1, $s1
mfc1	Move uma word do C1 para um reg. de propósito geral	mfc1 $s1, $f1

17

Outras Instruções

OP	Descrição	Exemplo
nop	Nao executada nada	nop
syscall	Gera uma exceção de chamada ao sistema operacional	syscall

18

Registradores

- Apenas 32 registradores?
  - Poucos e rápidos!
- E os registradores $at (1) e $k0,$k1 (26,27)?
  - Montador e SO

Nome	# do registrador	Uso	Preservado na Chamada?
$zero	0	Constante 0	-
$k0, $k1	0-1	Sistema Operacional	Não
$v0,$v1	2,3	Retorno de funções	Não
$a0-$a3	4-7	Argumentos	Não
$t0-$t7	8-15	Temporários	Não
$s0-$s7	16-23	Salvos	Sim
$t8,$t9	24,25	Mais temporários	Não
$gp	28	Ponteiro global	Sim
$sp	29	Ponteiro de pilha	Sim
$fp	30	Ponteiro de quadro	Sim
$ra	31	Endereço de retorno	Sim

19

Registradores do Coprocessador 1

- 32 registradores – $f0 ~$f31 (32 bits cada);
- Um registrador de estado (condition flag);
  - 8 bits CF7 ~ CF0;
- Dois modos de operação:
  - Single precision 32 bits;
  - Double precision 64 bits;
    - Registradores pareados $f0-$f1, $f2-$f3, •••

20

Registradores do Coprocessador 0

- Quatro registradores usandos para controle de excessões (traps) e estado do processador;
- $8 – (vaddr) – endereço da instrução que executou uma exceção;
- $12 – (status) – máscara de interrupção e bits de abilitação;
- $13 – (cause) – tipo de exceção é bits pendentes de interrupção;
- $14 – (epc) – endereço da instrução que causou uma exceção;

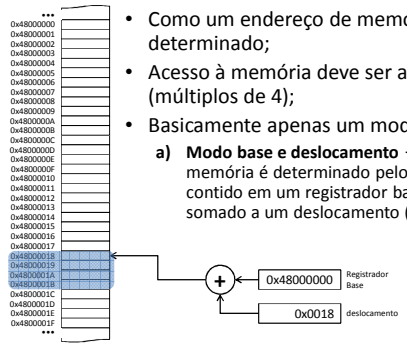
21

Outros Registradores

- Outros Registradores
  - **PC – Program Counter →** registrador que aponta para a próxima instrução a ser executada;
  - **IR – Instruction Register →** registrador que retêm a(s) instrução(ões) sendo executada(s);
  - **DR – Data Register →** contém o dado recuperado da memória
  - **HI – High →** contém os 32 bits mais significativos de uma multiplicação ou o resto de uma divisão inteira;
  - **LO – Low →** contém os 32 bits menos significativos de uma multiplicação ou o quociente de uma divisão inteira;

22

Modos de Endereçamento

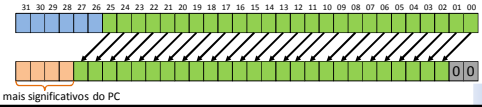


- Como um endereço de memória é determinado;
- Acesso à memória deve ser alinhado (múltiplos de 4);
- Basicamente apenas um modo!
  - **a) Modo base e deslocamento** → endereço de memória é determinado pelo endereço contido em um registrador base (32 bits) somado a um deslocamento (16 bits)

23

Modos de Endereçamento

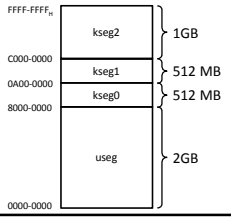
- Do ponto de vista funcional, podemos pensar em outros modos de endereçamento, se considerarmos também como referenciar registradores:
  - **b) Modo Registrador** → Apenas registradores são usados na instrução:
    - 5 bits são usados para diferenciar entre os 32 registradores de propósito geral (RPG);
    - Modo mais comum usado na arquitetura MIPS;
  - **c) Modo relativo ao PC** → usado em instruções de salto condicional:
    - Endereço destino é a soma do PC + deslocamento de 16 bits (sinalizado);
    - Deslocamento dado em palavras e deve ser multiplicado por 4;
    - Possível endereçar [-32.768, 32.767] palavras a partir do PC;
  - **d) Modo Imediato** → usado em instruções lógicas e aritméticas apenas:
    - A constante é um int16 [-32.768, 32.767];
  - **e) Modo absoluto** → Usado em desvios incondicionais:



24

Arquitetura de Memória

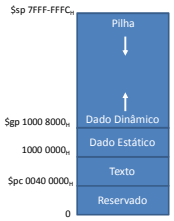
- Logicamente, a memória pode ser vista como um arranjo unidimencional de bytes individualmente endereçáveis;
- Organizacionalmente a memória é dividida em duas grandes regiões (kseg, e useg) que podem ser subdivididas em outras subregiões;



25

Abstração de Memória

Possível Organização



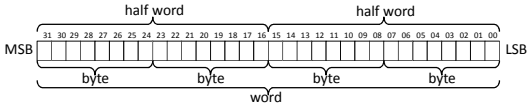
- Do ponto de vista do programador:
- 2^{32} endereços distintos
- 4.294.967.296 bytes
- 2.147.483.648 half words
- 1.073.741.824 words
- 34.359.738.368 bits
- 4 bytes por instrução
- Array de dados;
- Endereço identifica uma célula de memória individual;
- O conteúdo da célula contém efetivamente o dado/instrução;

Abstração Memória:

26

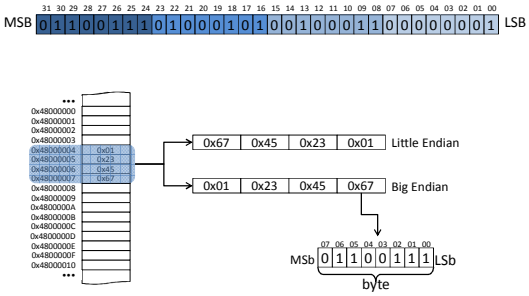
Ordenação de Bytes

- A únidade mínima endereçável é o byte;
- Uma palavra (word) possui 32 bits, ou seja, é composta por 4 bytes;
- Ordenação de bytes diz respeito a posição dentro de uma word dos bytes;
- Dois modelos principais:
  - Big endian
  - Little Endian



27

Ordenação de Bytes



28

Modos de Operação

- Originalmente µprocessadores possuiam apenas um modo de operação;
- Atualmente → pelo menos 2 modos são necessários;
  - Modo Kernel
  - Modo Supervisor
  - Modo de Usuário
  - Modo de Depuração

29

Referências

- Sweetman, D. See MIPS Run Linux. 2nd. Ed. Elsevier, 2007.

30