UNISUL

Arquitetura de Computadores

Prof. Fábio Eduardo Vieira Angelo
UNISUL – 2019

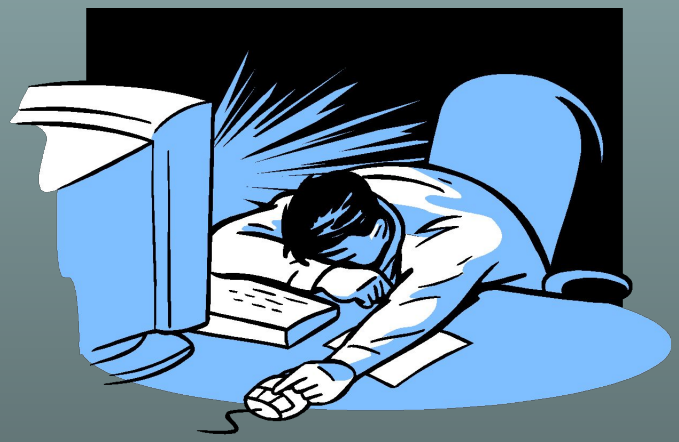
UNISUL

Como o computador se comunica?

UNISUL

Como o computador se comunica?

- Embora os dialetos possam mudar, todos os computadores usam o mesmo idioma:
  - BINÁRIO obtido através de dois níveis elétricos!



UNISUL

Revisão dos Sistemas Numéricos

- Antes de continuar nosso estudo, devemos fazer uma parada para um breve revisão.
- Diversos sistemas numéricos podem fazer parte do nosso cotidiano: decimal, binário, hexadecimal, octal, etc..
- Os mais importantes neste momento:
  - Binário
  - Decimal

UNISUL

Revisão Sobre Sistemas Numéricos

- A primeira comunidade que acrescentou à história uma herança hoje reconhecida como intelectual foi a dos sumérios, um povo que viveu na região onde hoje fica o Iraque.
- Intelectualidade é aquela capacidade que só os homens têm de perceber coisas fora do alcance dos seus cinco sentidos, o chamado pensamento abstrato.

UNISUL

Revisão Sobre Sistemas Numéricos

- Os sumérios foram os responsáveis por algumas abstrações notáveis, que turbinaram a marcha da humanidade e influenciaram para sempre o modo como as pessoas pensam, agem e se comunicam.
- Aos sumérios se deve a invenção da cerveja. Muita gente diria que, se eles não tivessem feito mais nada, só com isso já teriam garantido seu lugar no trem da história (e na janelinha).

UNISUL

Revisão Sobre Sistemas Numéricos

- Devem-se os sumérios a idéia de que os homens foram criados à imagem dos deuses, a fabricação dos primeiros instrumentos agrícolas, as tentativas iniciais de organização de cidades, os rudimentos do cooperativismo e a fabricação do vidro.
- Nesse momento o aluno pode se perguntar o que da intelectualidade ainda estava por vir: os sumérios foram a primeira civilização da Terra capaz de registrar a própria história, porque eles inventaram a PALAVRA ESCRITA. Isto ocorreu 2300 anos antes de Cristo.

UNISUL

Revisão Sobre Sistemas Numéricos

- Foi deles, há 5 mil anos, a idéia de criar símbolos que pudessem representar os sons vocais, através de uma escrita chamada cuneiforme. O nome pode ser complicado, mas a idéia era bem simples: marcas feitas em tabletes úmidos de barro, que depois eram secados ao sol. Os estiletes de madeira usados para marcar os símbolos no barro tinham a ponta em forma de cunha ou eram cuneiformes.

UNISUL

Revisão Sobre Sistemas Numéricos

- Hoje, pode parecer que passar da palavra escrita para os algarismos numérico foi um pequeno passo, quase uma conseqüência natural, mas entre os dois fatos atravessamos um abismo de alguns séculos. Muita gente pode não ter esta impressão, mas aprender a escrever é muito mais simples que aprender a calcular. Tanto que, enquanto a escrita se desenvolvia através de diferentes símbolos para diferentes sílabas, as contagens continuavam a ser feitas com base no conceito 1. Um evento, igual a um risquinho, uma pedra, um osso, etc.

UNISUL

Revisão Sobre Sistemas Numéricos

- Muitas categorias de profissionais contribuíram para o avanço da ciência do cálculo e uma delas foi a dos primitivos pastores. Durante séculos a fio, eles foram repetindo uma mesma e modorrenta rotina, a de soltar os seus rebanhos pela manhã, para pastar em campo aberto, e de recolhe-los à tardinha. Tudo na maior tranqüilidade, até que alguém levantou a questão:
  - ***COMO VOCÊ SABE SE A QUANTIDADE DE OVELHAS QUE SAIU FOI A MESMA QUE VOLTOU?***

UNISUL

Revisão Sobre Sistemas Numéricos

- Problema seriíssimo, de fato, e que foi solucionado rapidamente para os padrões da época, ou seja, em menos de um milênio por um iluminado pastor que resolveu a situação da seguinte maneira: pela manhã, ele fazia um montinho de pedras, colocando nele uma pedra para cada ovelha que saía; e, à noite, retirava uma pedra para cada ovelha que voltava. O número de ovelhas desgarradas correspondia a quantidade de pedras restantes no monte. Mesmo sem saber, aquele pastor foi o primeiro ser humano a calcular. Porque **PEDRA,** em latim, é **CALCULUS**. Que por isso também é a origem da expressão cálculo renal (pedra no rim).

UNISUL

Revisão Sobre Sistemas Numéricos

- **Mas Calcular Era Tão Complicado Assim?**
- Muito! A primeira maneira que o seres humanos encontram para mostrar a que quantidade estavam se referindo foi o uso dos dedos da mão. Hoje isto pode parecer brincadeira, mas 5 mil anos atrás para contar até 20 eram necessários dois homens, porque tinham que ser usadas quatro mãos.

UNISUL

Revisão Sobre Sistemas Numéricos

- Demorou alguns séculos até que alguém se desse conta e dissesse:
  - **OLHA PESSOAL, JÁ ACUMULEI O RESULTADO DE DUAS MÃOS E AGORA VOU CONTINUAR VOLTANDO À PRIMEIRA MÃO!**
- Através dos tempos, as mãos foram sendo substituídas por equipamentos mais sofisticados, mas a palavra em latim para dedo, **DIGITUS**, sobrevive até hoje, tanto na palavra dígito (um algarismo), quanto no verbo digitar (escrever com os dedos).

UNISUL

Revisão Sobre Sistemas Numéricos

- Mas mostrar os dedos era uma coisa e saber contar era outra. A maioria dos povos da idade do dedo sabia contar apenas até três; do quatro em diante a coisa já entrava em uma dimensão fantástica. Em praticamente todos os idiomas, as palavras para os três primeiros algarismos se parecem, porque todas elas derivam de uma língua antiqüíssima e já extinta, o *sânscrito*: an, dve, dri. Já a noção de quatro é tão posterior que cada povo desenvolveu sua própria palavra para expressa-la.

UNISUL

Revisão Sobre Sistemas Numéricos

- **QUANDO A HUMANIDADE APRENDEU A CONTAR?**
  - Há cerca de 4 mil anos. Foi quando os mercadores da Mesopotâmia desenvolveram o primeiro sistema científico para contar e acumular grandes quantias. Primeiro eles faziam um sulco na areia e iam colocando nele sementes secas (ou contas) até chegar a dez. Aí faziam um segundo sulco, onde colocavam uma só conta - o que equivalia a 10 - esvaziavam o primeiro sulco e repetiam a operação: cada 10 contas no primeiro sulco valia uma conta no segundo sulco.

UNISUL

E o Sistema Binário?

- É o sistema de numeração mais utilizado em processamento de dados digitais, pois utiliza apenas dois algarismos (0 e 1), sendo portanto mais fácil de ser representado por circuitos eletrônicos (Os dígitos binários podem ser representados pela presença ou não de tensão). O sistema de numeração binário utiliza a base 2. Cada posição de cada algarismo de um número binário corresponde a uma potência de 2, analogamente ao sistema decimal, onde cada posição corresponde a uma potência de dez.

UNISUL

Sistema Binário

- Os dígitos binários chamam-se bits, proveniente da contração do inglês Binary Digit. Assim como no sistema decimal, dependendo do posicionamento, o algarismo ou bit terá um peso. O da extrema esquerda será o bit mais significativo (Most Significant Bit - MSB) e o da extrema direita o bit menos significativo (Least Significante Bit - LSB). Um conjunto de 8 bits é denominado byte. Um conjunto de 4 bits é denominado nybble.

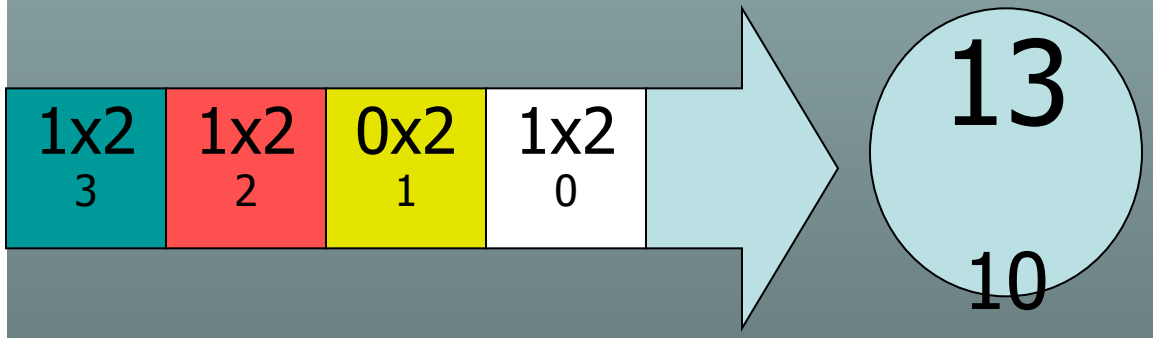
UNISUL

Sistema Binário

- Quando lidamos com um número binário, usualmente ficamos restritos a representá-los com certo número de bit´s. Está restrição, geralmente, está relacionada com ao circuito utilizado para a contagem.

Sistema Binário

- Conversão: 1101



UNISUL

Instruções Básicas de linguagem de Máquina

- Linguagem de Máquina instruções.
  - Pequenas diferenças entre diferentes máquinas.
  - Todos os processadores derivam da mesma tecnologia de hardware.
- Mais primitiva que linguagens de alto nível, o que leva a um controle de fluxo não sofisticado.
- Muito restritiva
  ex. MIPS Instruções Aritméticas.
- Nós trabalharemos com a arquitetura do conjunto de instruções do MIPS .
  - similar a outras arquiteturas desenvolvidas após 1980.

UNISUL

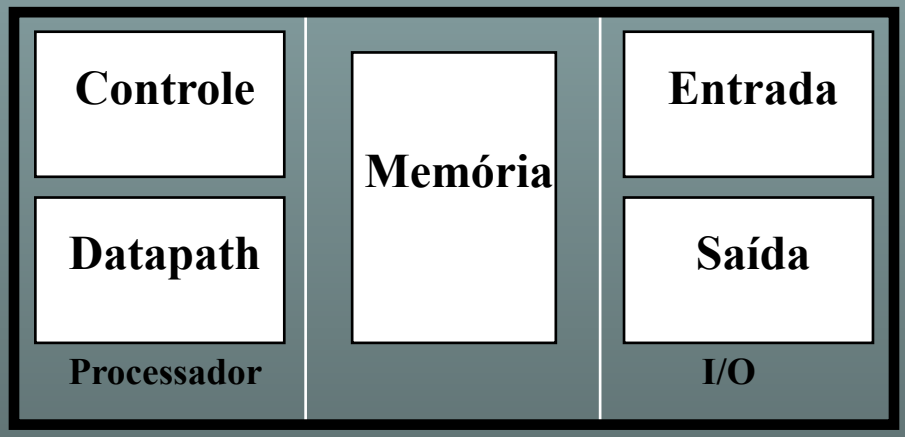
Instruções aritméticas

- O mínimo que se espera de um computador é que ele seja capaz de somar. Entretanto, em linguagem de máquina devemos considerar algumas restrições, com o número de operando:
  - Sempre três operandos.

Registradores x Memória

- Operandos de instruções Aritméticas devem ser registradores.
- Compilador associa variáveis com registradores.
- Como fazer quando um programa tem muitas variáveis?

O que fazer com Array?



UNISUL

MIPS - Aritmética

- Todas instruções tem 3 operandos
- A ordem dos operandos é fixa (destino primeiro)

Exemplo:

Código C: `A = B + C`
Código MIPS: `add $s0, $s1, $s2`

UNISUL

MIPS - Aritmética

- Principio de projeto: simplicidade favorece a regularidade.

- Código C: `A = B + C + D;`
  `E = F - A;`

  Código MIPS:
  ```
  add $t0, $s1, $s2
  add $s0, $t0, $s3
  sub $s4, $s5, $s0
  ```
  Operandos devem ser registradores, só existem 32 registradores.
- Palavras do MIPS com 32 bits.
- Princípios de Projeto:
  "A simplicidade é favorecidade pela regularidade."
  "Menor é mais rápido."

UNISUL

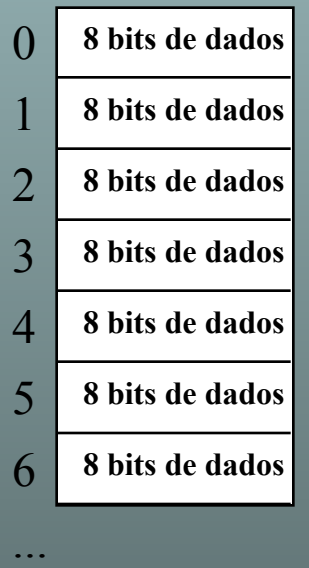
MIPS - Aritmética

- Devemos lembrar que as instruções vistas são representações simbólicas da linguagem que um processador realmente entende.

UNISUL

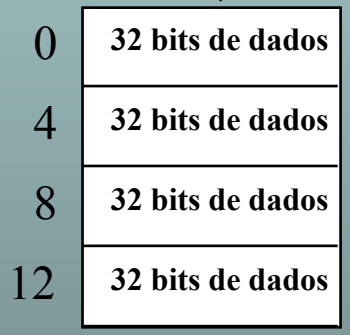
Organização de Memória

- Visto como uma matriz unidimensional, com um endereço.
- Um endereço de memória é um índice em uma matriz
- Endereçamento de Byte aponta para um byte da memória.



Organização da Memória

- Bytes são pequenos, mas vários dados usam "words" (palavra).
- Para o MIPS, uma palavra tem 32 bits ou 4 bytes.



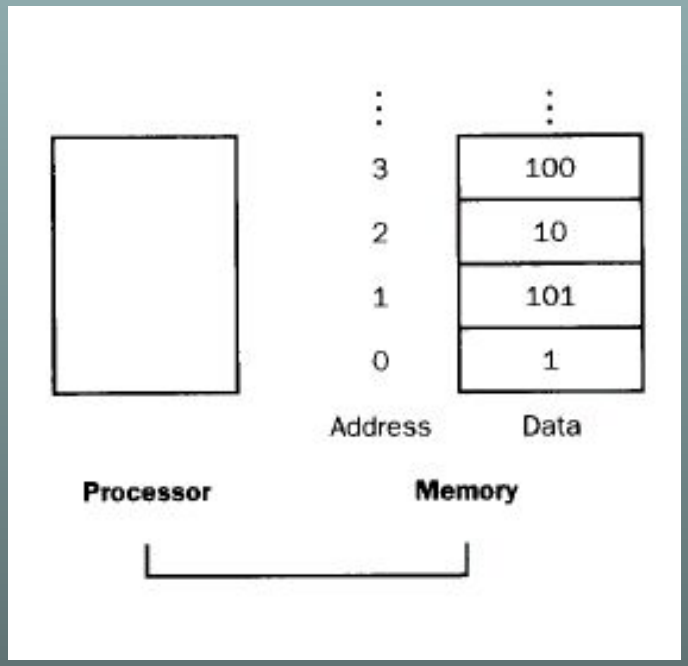
Registradores retém 32 bits de dados

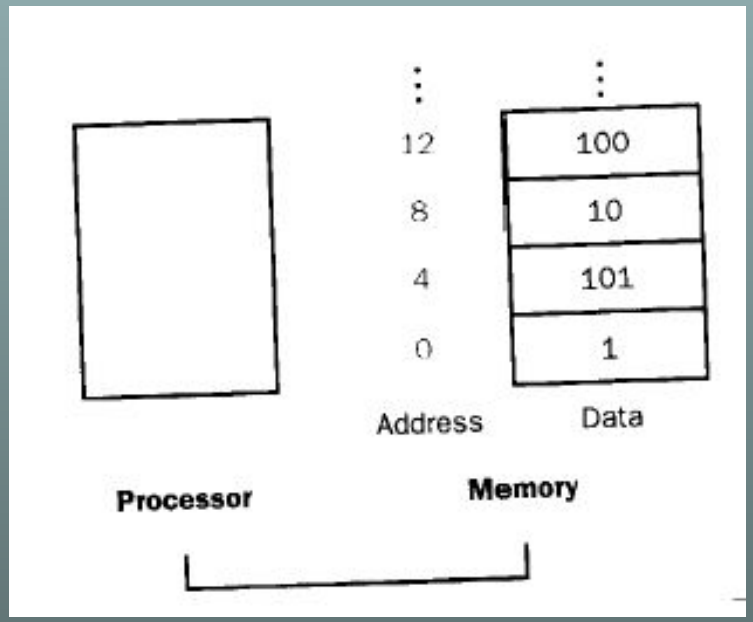
- 2^{30} palavras com endereço de byte 0, 4, 8, ... 2^{32}-4 □ estas posições só são acessadas por instruções de transferência de dados.

- Palavras são alinhadas - RESTRIÇÃO DE ALINHAMENTO.

Endereçamento: Bytes x Palavras

- Endereçamento Byte
- Endereçamento Palavra





UNISUL

Transferência de Dados Entre Memória e Registrador

- Para transferir uma palavra de dados, nós devemos especificar duas coisas:
  - Registrador: especifique este pelo número (0 - 31)
  - Endereço da memória: mais difícil.
- Pense na memória como um array único e unidimensional, de modo que podemos endereçá-la simplesmente fornecendo um ponteiro para um endereço da memória.

UNISUL

Instruções Utilizando Conteúdo de Memória

- Instruções load e store.

Código C: `A[8] = h + A[8];`

Código MIPS:
```
lw $t0, 32($s3)
add $t0, $s2, $t0
sw $t0, 32($s3)
```

- Store tem destino por último.
- Lembre-se de que operandos aritméticos são registradores, NÃO MEMÓRIA!

UNISUL

Exemplo: Compilar ...

- Compilar manualmente usando registradores:
  - A[12] = h + A[8];
  - h: $s2, $s3: endereço base de A
- 1- transfere da memória para registrador $t0:
  - lw $t0,32($s3) # $t0 = A[8]
- 2 - some-o a h e coloque em $t0:
  - add $t0,$s2,$t0 # $t0 = h+A[8]
- 3 - transfere do reg. $t0 para a memória :
  - sw $t0,48($s3) # A[12] = $t0

UNISUL

Notas Sobre Memória

- Falha: Esquecer que endereços seqüenciais de palavras em máquinas com endereçamento de byte não diferem por 1.
  - Muitos erros são cometidos por programadores de linguagem assembly por assumirem que o endereço da próxima palavra pode ser achado incrementando-se o endereço em um registrador por 1 ao invés do tamanho da palavra em bytes.
  - Logo, lembre-se que tanto para lw e sw, a soma do endereço base e o offset deve ser um múltiplo de 4 (para ser alinhado em palavra)

UNISUL

Papel dos Registradores x Memória

- E se temos mais variáveis do que registradores?
  - Compilador tenta manter as variáveis mais freqüentemente utilizadas nos registradores.
  - Escrevendo as menos comuns na memória: SPILLING.
- Por que não manter todas as variáveis na memória?
  - Devido ao princípio Menor é mais rápido: registradores são mais rápidos que a memória.
  - Registradores são mais versáteis:
    - Instruções aritméticas MIPS pode ler 2, operar sobre eles e escrever 1 por instrução.
    - Transferência de dados MIPS somente lê ou grava 1 operando por instrução, sem nenhuma operação.

UNISUL

Conclusões Parciais

- Em linguagem Assembly MIPS:
  - Registradores substituem variáveis C.
  - Uma instrução (operação simples) por linha.
  - Mais Simples é Melhor.
  - Menor é Mais Rápido.
- Memória é endereçada por byte, mas lw e sw acessam uma palavra de cada vez.

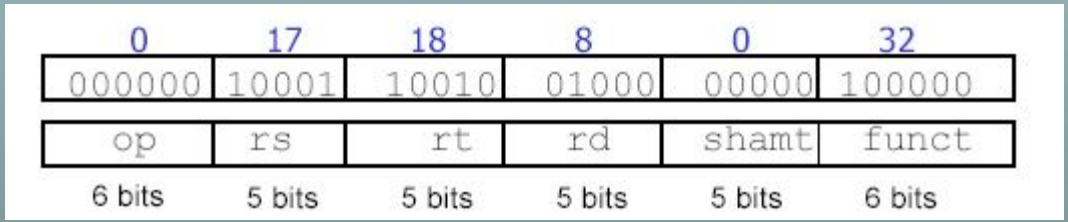
UNISUL

Representação de Instruções

- Os números são mantidos no computador como um conjunto de sinais eletrônicos que podem assumir apenas dois estados : alto e baixo.
- Por isso todas as informações em linguagem de máquina são entendidas como zeros e uns.
- Na linguagem do MIPS os nomes $s0 a $s7 são mapeados nos registradores 16 a 23, e os nomes $t0 a $t7 nos registradores 8 a 15.

Linguagem de Máquina

- Instruções, como registradores e palavras, são de 32 bits.
  - Exemplo: add $t0, $s1, $s2.
  - registradores tem números t0=reg.8,$s1=reg.17,$s2=reg.18
- Formato de Instrução de soma com registradores (tipo R):



- Primeiro campo: tipo de operação (soma).
- Último campo: modo da operação (soma com 2 registradores).
- Segundo campo: primeira fonte (17=$s1).
- Terceiro campo: segunda fonte (18=$s2).
- Quarto campo: registrador de destino (8=$t0).

UNISUL

Linguagem de Máquina

- Novo princípio: Bom projeto exige um bom compromisso.
  - Vários tipos de instruções (tipo-R, tipo-I).
  - Múltiplos formatos: complicam o hardware.
  - Manter os formatos similares (3 primeiros campos iguais).
- Considere as instruções load-word e store-word,
  - I-tipo para instruções de transferência de dados (lw,sw)
- Exemplo: lw $t0, 32($s3).

35	19	8	32
op	rs	rt	16 bit - offset

- Registrador de base: rs $s2.
- Registrador de fonte ou origem: rt $t0.

UNISUL

Codificação das Instruções Vistas

Instrução	Formato	op	rs	rt	rd	shamt	funct	address
add	R	0	reg	reg	reg	0	32	n.a.
sub	R	0	reg	reg	reg	0	34	n.a
lw	I	35	reg	reg	n.a.	n.a.	n.a.	Endereço
sw	I	43	reg	reg	n.a.	n.a.	n.a.	Endereço

- Notar: add e sub:
  - Mesmo opcode: 0.
  - Diferente função: 32 e 34.

Exemplo

- Tradução de C para assembly e linguagem de máquina.
  - C: A[300] = h + A[300]
  - Assembly: # $t1= base A

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

  - Linguagem de máquina:

"simplicidade favorece regularidade"

1 bit de diferença

op	rs	rt	rd	address/ shamt	funct
35	9	8	1200		
0	18	8	8	0	32
43	9	8	1200		
$10^0$011	01001	01000	0000 0100 1011 0000		
000000	10010	01000	01000	00000	100000
$10^1$011	01001	01000	0000 0100 1011 0000		

Resumo Geral da Linguagem Vista

Name	Example	Comments
32 registers	$s0, $s1, . . . , $s7 $t0, $t1, . . . , $t7	Fast locations for data. In MIPS, data must be in registers to perform arithmetic. Registers $s0–$s7 map to 16–23 and $t0–$t7 map to 8–15.
2^{30} memory words	Memory[0], Memory[4], . . . , Memory[4294967292]	Accessed only by data transfer instructions in MIPS. MIPS uses byte addresses, so sequential words differ by 4. Memory holds data structures, such as arrays, and spilled registers.

MIPS assembly language

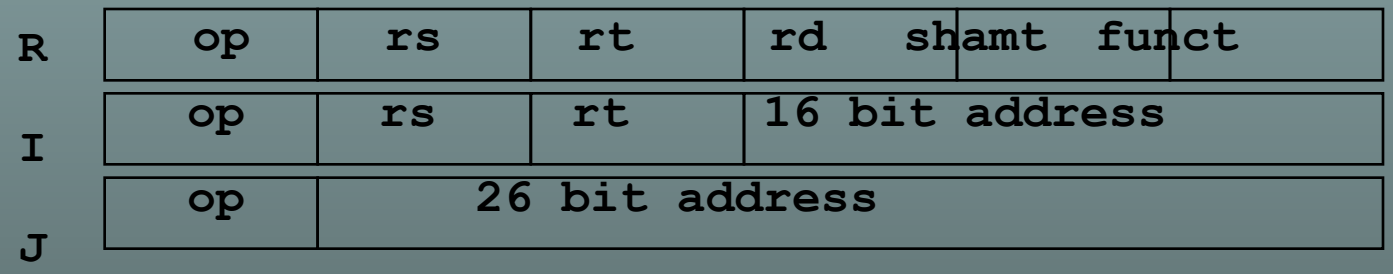
Category	Instruction	Example	Meaning	Comments
Arithmetic	add	add $s1,$s2,$s3	$s1 = $s2 + $s3	Three operands; data in registers
	subtract	sub $s1,$s2,$s3	$s1 = $s2 - $s3	Three operands; data in registers
Data transfer	load word	lw $s1,100($s2)	$s1 = Memory[$s2 + 100]	Data from memory to register
	store word	sw $s1,100($s2)	Memory[$s2 + 100] = $s1	Data from register to memory

MIPS machine language

Name	Format	Example						Comments
add	R	0	18	19	17	0	32	add $s1,$s2,$s3
sub	R	0	18	19	17	0	34	sub $s1,$s2,$s3
lw	I	35	18	17	100			lw $s1,100($s2)
sw	I	43	18	17	100			sw $s1,100($s2)
Field size		6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	All MIPS instructions 32 bits
R-format	R	op	rs	rt	rd	shamt	funct	Arithmetic instruction format
I-format	I	op	rs	rt	address			Data transfer format

Instruções de Desvios

- Assim como todo computador, o MIPS precisa tomar decisões:
  - if
  - goto
  - if-else
  - while



UNISUL

Compilando um “if”

- O código C é:
  - if(i==j) go to L1
  - f=g+h
  - L1: f=f-i
- O código assembly:
  - beq $s3, $s4, L1 # desvia para L1 se i=j
  - add $s0, $s1, $s2 # f=g+h
  - L1: sub $s0,$s0,$s3 # f=f-i (sempre executada)

    - Considerar as variáveis (f, g, h, i e j) associadas aos registradores $s0 a $s4.

Interface Hardware / Software do “if”

- Com freqüência os compiladores criam desvios e labels que não aparecem na linguagem de alto nível.
  - Evitar tal desgastes é uma das grandes vantagens das linguagens de alto nível.

UNISUL

Compilando um if-then-else

- O código C é:
  - if(i==j) f=g+h
  - else f=g-h
- O código MIPS:
  - bne $S3, $S4, ELSE # desvia para ELSE se i≠j.
  - add $S0, $S1, $S2 # f=g+h
  - j EXIT # desvia para EXIT
  - ELSE: sub $S0, $S1, $S3 # f=g+h (salta esta se i=j)
  - EXIT

UNISUL

Política de Uso de Convenções

Name	Register number	Usage
$zero	0	the constant value 0
$v0-$v1	2-3	values for results and expression evaluation
$a0-$a3	4-7	arguments
$t0-$t7	8-15	temporaries
$s0-$s7	16-23	saved
$t8-$t9	24-25	more temporaries
$gp	28	global pointer
$sp	29	stack pointer
$fp	30	frame pointer
$ra	31	return address

Decodificação de Linguagem de Máquina

- Qual a instrução em linguagem assembly para a seguinte instrução de máquina?
  - 0000 0000 1010 1111 1000 0000 0010 0000

op	rs	rt	rd	Shamt	funct
000000	00101	01111	10000	00000	100000

  - Add $s0,$a1,$t7

UNISUL

Além dos Números

- Os computadores foram inventados para manipular números, mas com a popularização passaram a processar textos. A grande maioria das máquinas usa um byte para representar caracteres internamente, utilizando o código ASCII.

UNISUL

80x86

- 1978: Intel 8086 é anunciado (arquitetura 16 bit)
- 1980: 8087 coprocessador ponto flutuante é incluído
- 1982: 80286 melhora espaço de endereçamento 24 bits, +instruções
- 1985: 80386 estende para 32 bits, novos modos de endereçamento
- 1989-1995: 80486, Pentium, Pentium Pro adicionou-se novas instruções (projetado para alta performance)
- 1997: MMX aparece

"Esta história ilustra o impacto da manutenção da compatibilidade"
"Um arquitetura que é difícil explicar e impossível amar"

UNISUL

80x86

- Algumas vezes, os projetistas dos conjuntos de instruções optam por oferecer, em suas instruções, operações mais poderosas que as do MIPS, com o objetivo de reduzir o número de instruções executadas por um programa. O perigo é que essa redução no número de instruções possa reduzir a simplicidade, aumentando o tempo de execução em função de instruções mais lentas.

UNISUL

80x86

- O 80x86 possui apenas 8 registradores de uso geral.
  - Os programas no MIPS podem usar quatro vezes mais registradores.
- As instruções aritméticas, lógicas e de transferência de dados possuem apenas dois operandos, forçando a situação em que um operando atue como fonte e destino.

UNISUL

Considerações Finais

- A simplicidade é favorecida pela regularidade.
- Quanto menor, mais rápido.
- Um bom projeto demanda compromisso.
- Torne o caso mais comum mais rápido.

UNISUL

FIM

- A comunicação com a máquina digital pode não ser tão simples quanto se imagina.