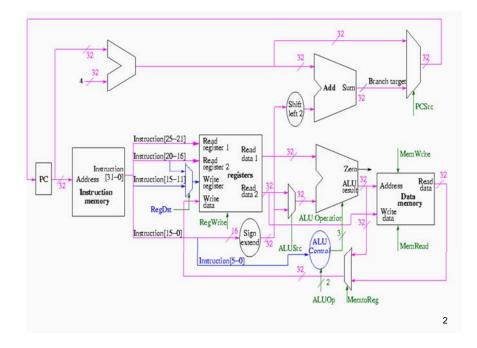
Caminho de dados – Datapath



Branch

RegDst

MemRead

MemUsce

AUUSc

RegWite

Instruction[25-21]
Instruction

Address [31-0]
Instruction

Instruction[15-1]
Instruction

Mem Write

AUUSc

Registers

Read

data 1
register

Write

data 2

Minemory

Autu Operation

Autu Operation

Autu Operation

Autu Operation

Sum

Autu Operation

Autu Operation

Sum

Autu Operation

Autu Operation

Sum

Autu Operation

Autu



- O desempenho de uma máquina pode ser determinado por três fatores:
 - número de instruções executadas
 - período do clock (ou frequência)
 - Número de ciclos por instrução (CPI)
- O compilador e a ISA (Instruction Set Arquitecture) determinam a quantidade de instruções a serem executadas por certo programa



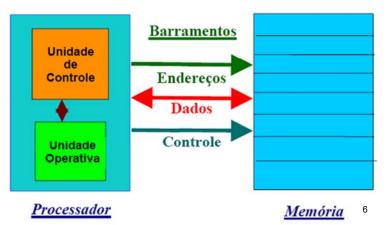
- Este capitulo aborda a implementação de um subconjunto das instruções do MIPS
- O interrelacionamento entre ISA e a implementação é exemplificado em dois projetos alternativos da parte operativa do processador
 - PO uniciclo
 - PO multiciclo

5



Arquitetura Von Neumann

Estrutura básica de um processador





Unidade Operativa

- Também chamada "Parte Operativa", "Via de Dados" ou, em inglês, "Datapath"
- É construída a partir dos seguintes componentes:
 - elementos de armazenamento (registradores, ffs)
 - operadores lógico-aritméticos
 - recursos de interconexão (barramentos, mux)



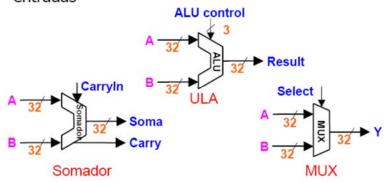
Unidade Operativa MIPS

- Será projetada para implementar o seguinte subconjunto de instruções do MIPS:
 - Instruções de referência a memória:
 - load word (lw) e store word (sw)
 - Instruções Aritméticas e lógicas:
 - add, sub, and, or e slt
 - Instruções de desvio de fluxo:
 - equal (beq), jump (j)



Componentes Combinacionais

 Componentes combinacionais definem o valor de suas saídas apenas em função dos valores presentes nas suas entradas



Write Enable

Data In

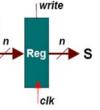
Address

٤



Componentes Sequenciais

- Componentes sequenciais tem um estado, que define seu valor em um dado instante de tempo
- Registrador:
 - Um conjunto de flip-flops tipo D (Registrador)
 - Com n bits de entrada e saída
 - entrada de habilitação de escrita (write enable)
 - Habilitação de escrita (write enable):
 - falso (0): O dado de saída não muda
 - verdade (1): o dado de entrada será carregado (saída = entrada)



10

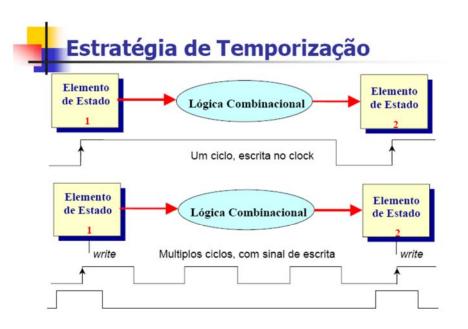
Memória

- Memória (idealizada)
 - Um barramento de entrada: Data In
 - Um barramento de saída: Data Out
- Uma palavra é selecionada por:
 - Um endereço seleciona a palavra para ser colocada na saída (*Data out*)
 - Write Enable = 1: Permite que a palavra selecionada seja escrita com a informação na entrada (Data in)
- Entrada de Clock (CLK)
 - Sincroniza os processos de acesso à memória
 - Geralmente, os processos s\u00e3o sincronizados pela borda de subida ou de descida do clock



Estratégia de Temporização

- Uma metodologia de temporização define quando os sinais podem ser lidos e quando eles podem ser escritos
- É necessário evitar situações de conflito, por exemplo, querer ler uma palavra e escrevê-la simultaneamente
- Será adotada uma metodologia de temporização sensível às transições do sinal do clock
- Nesta metodologia, qualquer valor armazenado nos elementos de estado só pode ser atualizado durante a transição do sinal de relógio (clk)



Criand

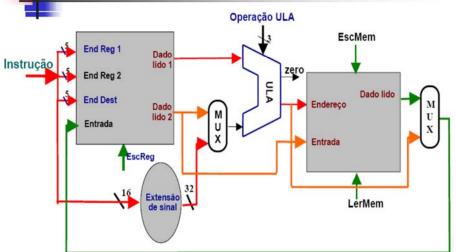
Criando a Unidade Operativa

- Uma maneira de se começar o projeto de uma unidade operativa (data path) é examinar cada um dos componentes necessários para a execução de cada uma das classes de instruções do MIPS
- Elementos necessários:
 - um lugar para armazenar as instruções (memória de instruções)
 - Um registrador para armazenar o endereço de instrução (*Program Counter PC*)
 - Um contador para incrementar o PC, para compor o endereço da próxima instrução (soma 4 para endereçar próxima instrução)

13

14

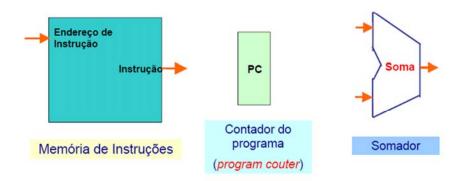
Criando a Unidade Operativa ...



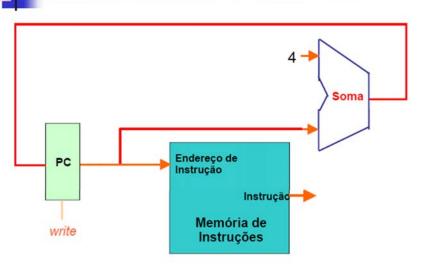
4

Criando a Unidade Operativa ...

Elementos Básicos



Busca de Instruções



Instruções Lógico-Aritméticas

Formato de uma instrução tipo R no MIPS:

ор	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

- Semântica:
 - \$rd <- op(\$rs, \$rt)</p>
- Estrutura de suporte: banco de registradores

17

19

-

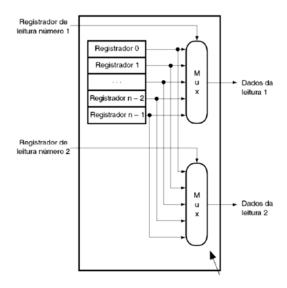
Banco de Registradores

- Dupla porta: leitura de dois registradores ao mesmo tempo
- Sinal de controle para escrita leitura n\u00e3o necessita controle

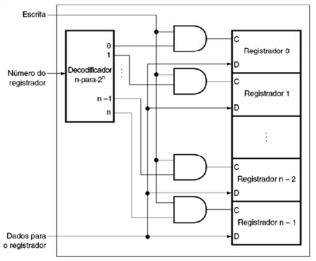


4

Banco de Registradores





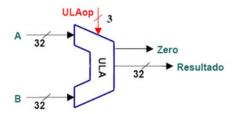




Projeto da ULA

- A ULA foi desenvolvida no capítulo anterior
- 3 bits de controle indicam a operação a ser realizada

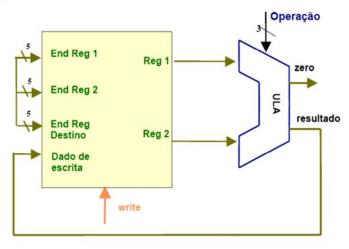
ULAop	Função
000	and
001	or
010	add
110	sub
111	slt



22



Instruções Tipo R - unid. operativa





Instruções de Acesso a Memória

■ Formato da instrução tipo I (lw/sw):

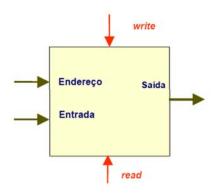
ор	rs	rt	endereço
6 bits	5 bits	5 bits	16 bits

- Ex:
 - lw \$8, 32(\$19)
 - \bullet end = \$19 + 32

35	19	8	32	
----	----	---	----	--



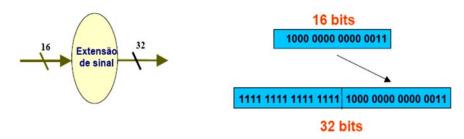
- Memória com um barramento de entrada independente do de saída
- Controle de escrita (write) e leitura (read)
- Barramento de endereços
- Um acesso de cada vez



25

Extensão de Sinal do Deslocamento

- Deslocamento na instrução deve ser estendido de 16 para 32 bits, mantendo-se o sinal
 - se for negativo, 16 bits superiores = 1
 - se for positivo, 16 bits superiores = 0



26

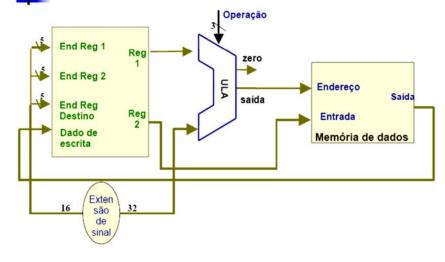


Acesso a Memória

- lw lê um dado da memória e escreve em um registrador
 - conexão entre a saída da memória e a entrada do banco de registradores
- sw lê um dado de um registrador e escreve na memória
 - ligação entre saída do banco de registradores e entrada de dados da memória
- endereço calculado através da ULA
 - saída da ULA ligada ao barramento de endereços da memória

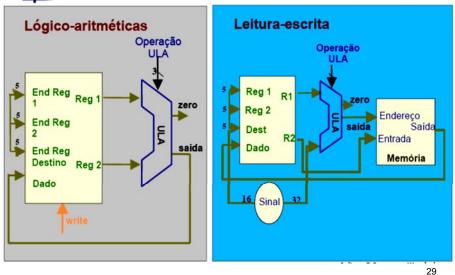
4

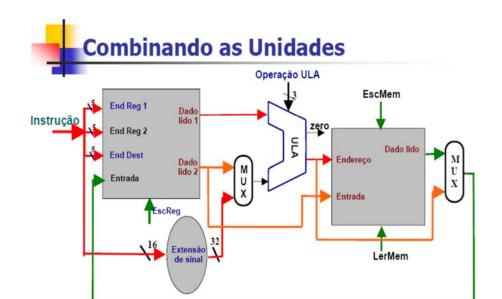
Unidade Operativa lw/sw





Instruções Lóg/Arit e lw/sw

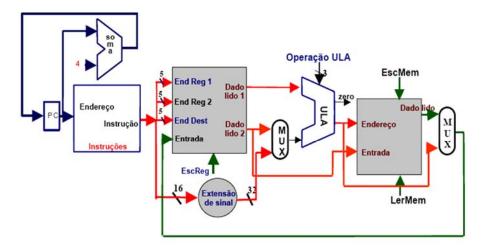




30



Acrescentando a Busca





31

Instruções de Desvio

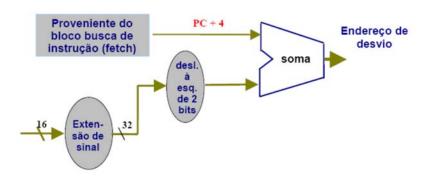
- beq \$1, \$2, desloc
 - compara dois registradores
 - soma desloc a PC+4 se \$1 = \$2
 - PC + 4 e o endereço da próxima instrução
 - no montador utiliza-se uma versão simplificada, com o rótulo do destino
 - beq \$1, \$2, Rótulo
 - neste caso, o montador calcula o deslocamento
 - desloc é um deslocamento de palavras, ou seja, cada unidade de desloc corresponde a 4 bytes

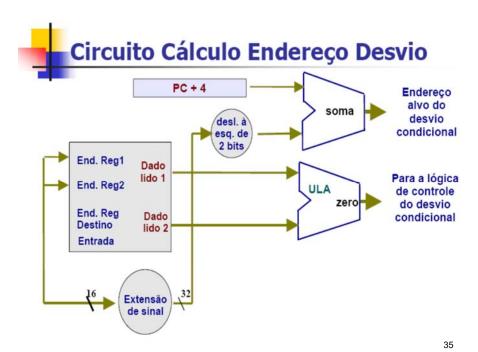


Instruções de Desvio

- A comparação entre os registradores é feita subtraindo-os na ULA e verificando se o resultado é zero
- O PC deve ser carregado com PC + 4 ou PC + 4 + desloc*4, de acordo com o resultado do teste
- A multiplicação de desloc por 4 é feita deslocando-se de 2 bits o seu valor









33

Observações

- Para realizar a comparação dos dois operandos precisamos usar o banco de registradores (os dois operandos estão lá)
- O cálculo do endereço de desvio foi incluído no circuito (já estudado)
- Na instrução não é preciso escrever no banco de registradores
- A comparação será feita pela ULA, subtraindose os registradores e utilizando a saída <u>zero</u> para verificar a igualdade



- Foi desenvolvido, na verdade, três tipos de unidades operativas:
 - uma para instruções no formato R (add, sub, etc.)
 - uma para instruções de no formato I (load e store)
 - uma para instruções condicionais (formato I)
- Na fase de projeto as vezes precisamos replicar recursos
- A via de dados mais simples deve-se propor executar as instruções num único período do clock
- Isto quer dizer que nenhum dos recursos pode ser usado mais de uma vez por instrução

Final

So M U X

So M U X

End Reg 1

Dado lido 1

Endereço listrução

Instrução

Instruções

End Reg 2

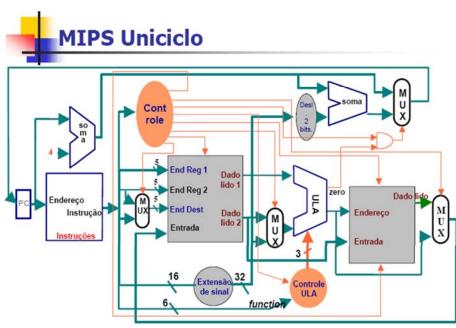
Ler Mem

Dado lido 2

Entrada

Ler Mem

37



Classes de instruções - tipo-R, load, store e branch)

Field	0	rs	rt	rd	shamt	funct
Bit positions	31-26	25-21	20-16	15-11	10-6	5-0
R-type instruction	on					
Field	35 or 43	rs	rt	or of 12 Hills o	address	
Bit positions	31-26	25-21	20-16	e sederitor (a	15-0	Halo
b. Load or store in	struction					
Field	4	rs	rt		address	
Bit positions	31-26	25-21	20-16		15-0	30100

Classes de instruções - tipo-R, load, store e branch)

Field	0	rs	rt	rd	shamt	funct
Bit positions	31-26	25-21	20-16	15-11	10-6	5-0
a. R-type instruction	on	en re et heid ditu				
Field	35 or 43	rs	rt	on the or on a	address	
Bit positions	31-26	25-21	20-16	e sederikui (se	15-0	
b. Load or store in	struction	en land to the first				
Field	4	rs	rt		address	
Bit positions	31-26	25-21	20-16		15-0	100000

Unidade de Controle Uniciclo

 A unidade de controle deve, a partir do código da instrução, fornecer os sinais que realizam as instruções na unidade operativa



42

4

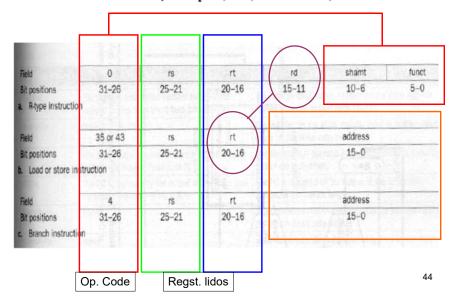
Controle do Add

- Por exemplo, a execução da instrução
 - add \$t1, \$s0, \$s2

requer as seguintes tarefas:

- encaminhar para a ULA o conteúdo dos registradores \$s0 e \$s2
- indicar para a ULA que vai ser realizada uma operação da adição
- Encaminhar o resultado para o registrado \$t1

Classes de instruções - tipo-R, load, store e branch)





Entrada da Unidade de Controle

- as informações necessárias a execução de uma instrução são retiradas da própria instrução
 - O opcode (código de operação) sempre está nos bits [31-26]
 - Os 2 registradores a serem lidos (rs e rt) sempre estão nas posições [25-21] e [20-16] (para todos os formatos!!!)
 - O registrador base (rs) para as instruções | w e sw sempre está especificado nas posições [25-21]
 - Os 16 bits de deslocamento para as instruções beq, lw e sw estão sempre nas posições [15-0]
 - O registrador destino está em uma das duas posições
 - [20-16] para lw (registrador rt)
 - [15-11] para instruções aritméticas/lógicas (registrador rd)

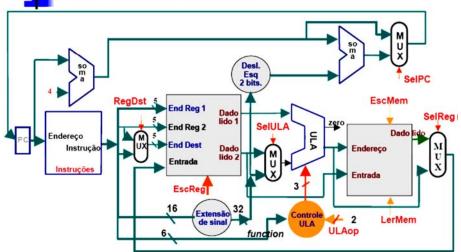
Sinais de Controle

- A unidade de controle de prover:
 - sinais para os multiplexadores
 - sinais de leitura e escrita para as memórias
 - seleção da operação da ULA
 - controle do novo endereço a ser carregado no PC, para instruções de salto

46

45

Sinais de controle





Instruções de Desvio

- A instrução de desvio condicional coloca:
 - PC+Desl ou (salto)
 - PC + 4 (instrução seguinte)

no contador de programa PC

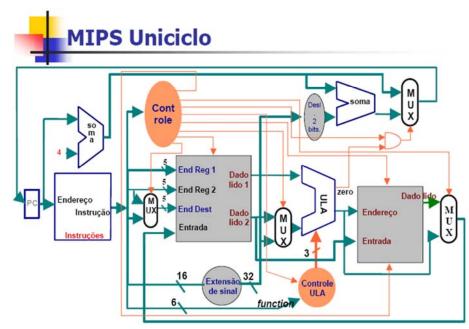
- É necessário selecionar SelPC em função:
 - do código da instrução (beq, bne)
 - do resultado da comparação (sinal zero da ULA)



Sinais de Controle

- LerMEM: dado da memória no endereço especificado é lido e colocado na saída
- EscMEM: conteúdo da memória endereçado recebe o dado
- SelULA:
 - 0 operando é a segunda saída do banco de registradores
 - 1 operando é o deslocamento extendido
- RegDST:
 - 0 índice do registrador destino é o campo rt da instrução
 - 1 índice do registrador destino é o campo rd da intrução
- EscREG: escreve dado no registrador indexado
- SelPC:
 - 0 PC recebe próximo endereço (PC+4)
 - 1 PC recebe endereço de desvio
- SelREG:
 - 0 registrador recebe saída da ULA
 - 1 registrador recebe saída da memória

40



50



Exercício

- Estender a organização do MIPS para dar suporte a execução de JUMP, desvio incondicional
- O endereço de desvio é obtido por:
 - PC[31 − 28] # Instrução[25 − 0] # 00
 - onde # indica concatenação de bits



Problemas com MIPS Uniciclo

- Período do relógio determinado pelo caminho mais longo
 - instrução lw:
 - leitura da instrução
 - leitura do registrador de base, extensão de sinal
 - cálculo do endereço
 - leitura do dado da memória
 - escrita em registrador
- TODAS as instruções levam o mesmo tempo para executar



- Supondo os seguintes tempos de execução das unidades do MIPS:
 - · Acesso a memória: 10 ns
 - ULA e somadores: 10 ns
 - Acesso ao banco de registradores: 5 ns
 - outros: 0 ns

Acesso a memória: 10 ns

• ULA e somadores: 10 ns

• outros: 0 ns

· Acesso ao banco de registradores: 5 ns

• Quais os tempos de execução das instruções supondo

seja, duração do ciclo igual a duração da instrução?

uma implementação uniciclo e outra com ciclo variável, ou

 Quais os tempos de execução das instruções supondo uma implementação uniciclo e outra com ciclo variável, ou seja, duração do ciclo igual a duração da instrução?

> Determinar o tempo de execução de cada instrução

lw = 10 + 5 + 10 + 10 + 5 = 40

sw = 10 + 5 + 10 + 10 = 35

tipo-R = 10 + 5 + 10 + 5 = 30

bea = 10 + 5 + 10 = 25

j = 10 Acesso a memória: 10 ns

ULA e somadores: 10 ns

Acesso ao banco de registradores: 5 ns

outros: 0 ns

 Quais os tempos de execução das instruções supondo uma implementação uniciclo e outra com ciclo variável, ou seja, duração do ciclo igual a duração da instrução?

Determinar o tempo de execução de cada instrução

lw =

sw =

tipo-R =

beq =

j =

Exemplo ...

- Considerando a distribuição de instruções do benchmark gcc, a diferença de velocidade entre as implementações seria:
 - GCC: 22% lw, 11% sw, 49% tipo-R, 16% beq, 2% jump
 - Período uniciclo: 40 ns
 - Período ciclo variável:

40*0.22 + 35*0.11 + 30*0.49 + 25*0.16 + 10*0.2 = 31.6 ns

Ganho: 40/31.6 = 1,27

55



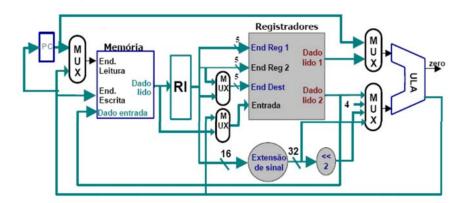
- Idéia: cada fase de execução de uma instrução dura um ciclo de relógio
- Instruções podem ser executadas em um número diferente de ciclos
 - lw leva 5 ciclos
 - jump leva 1 ciclo
 - add leva 4 ciclos



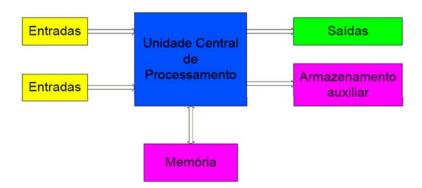
- Ciclo dimensionado de acordo com a fase mais demorada
- Unidades funcionais podem ser utilizadas para realizar mais de uma operação durante a execução de uma instrução
- A organização da parte operativa pode ser reestruturada em função destas características

57 58

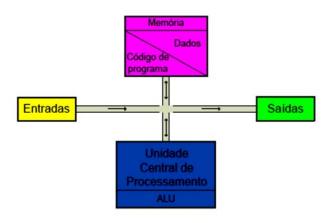




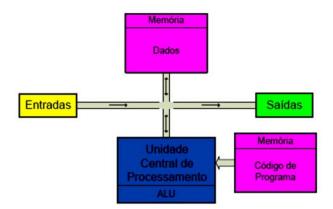
Arquitetura básica de um Microprocessador ou Microcontrolador



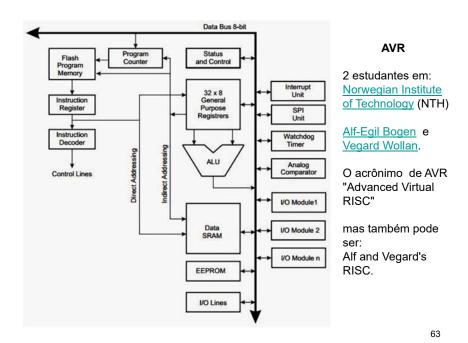
Arquitetura Von Neumann

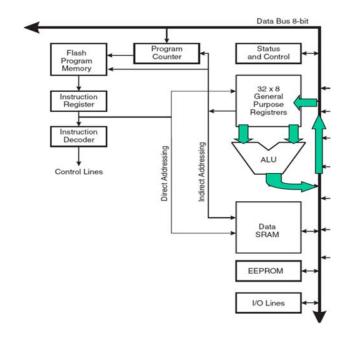


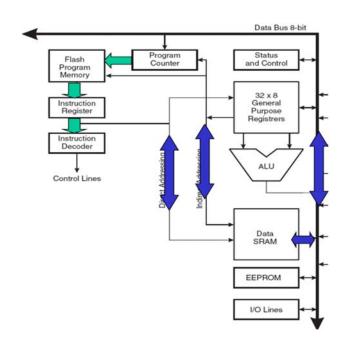
Arquitetura Harvard



61 62







10 ns 10 ns 10 ns Memória de Instruções 1 ns

Vetor com 100 elementos, achar o maior elemento, armazenar na última posição

- % (lw, sw, add, beq (ou bne), j) CPU time em us (f = 200 MHz)

lw	
sw	
add	
beq	
j	

67

65