UNIVERSIDADE FEDERAL DE RORAIMA

Barramenta - DataPath

Prof. Herbert Oliveira Rocha

UNIVERSIDADE FEDERAL DE RORAIMA

Barramenta - DataPath

Baseado nas aulas do Prof. Dr. Mauricio Figueiredo - UFAM

Prof. Herbert Oliveira Rocha

Introdução

- Objetivo: Construir um processador para execução de instruções.
  - Caminho de dados
  - Unidade de controle

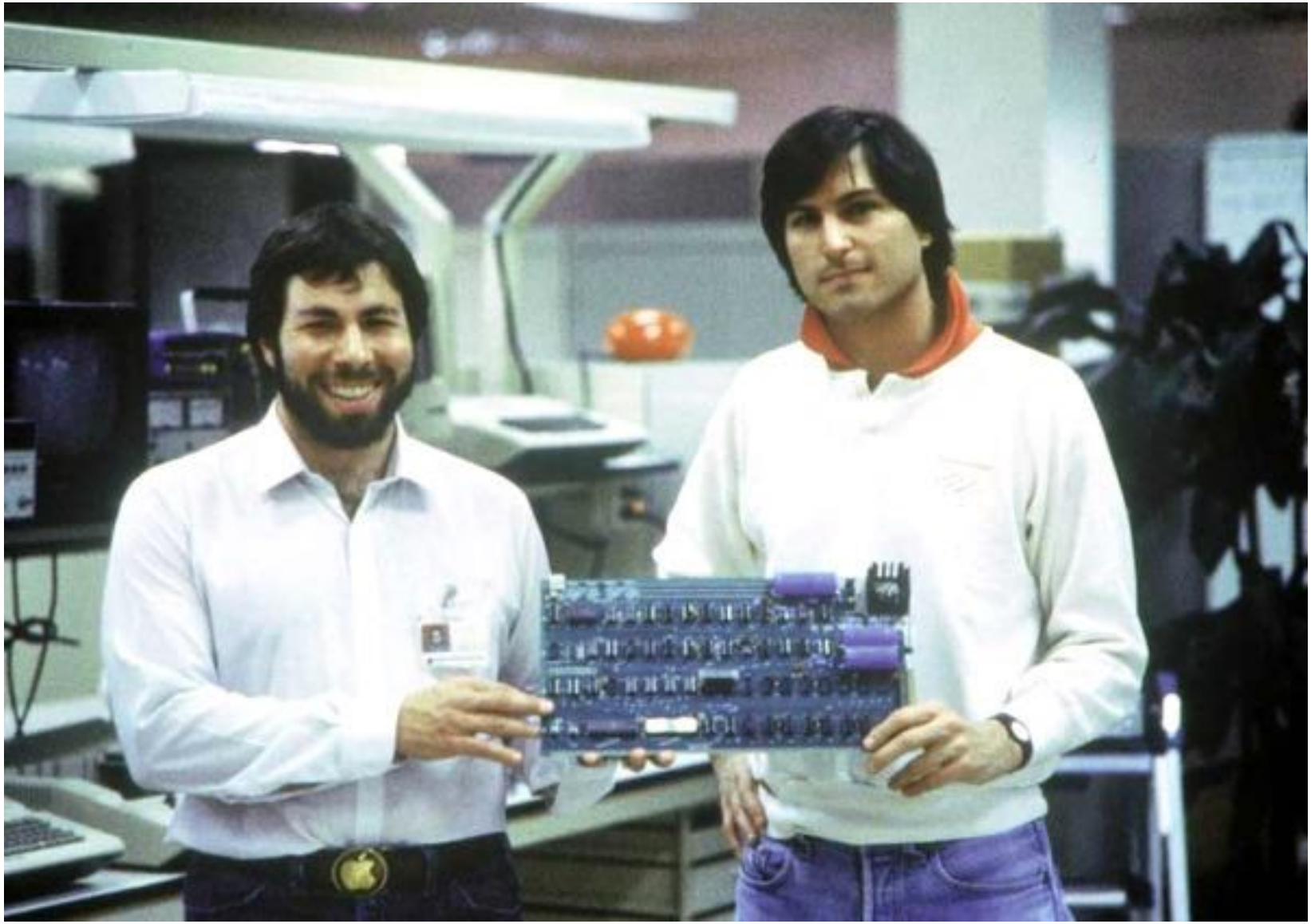
- O tempo do ciclo de clock e o número de ciclos por instrução são determinados pela implementação do processador.

- Projeto baseado no conjunto de instruções básicas do MIPS
  - Instruções de acesso à memória (lw e sw)
  - Instruções aritméticas (add, sub, and, or e slt)
  - Instruções de branch e jump (beq e j)

- Conjunto completo de instruções podem ser implementadas de forma similar.

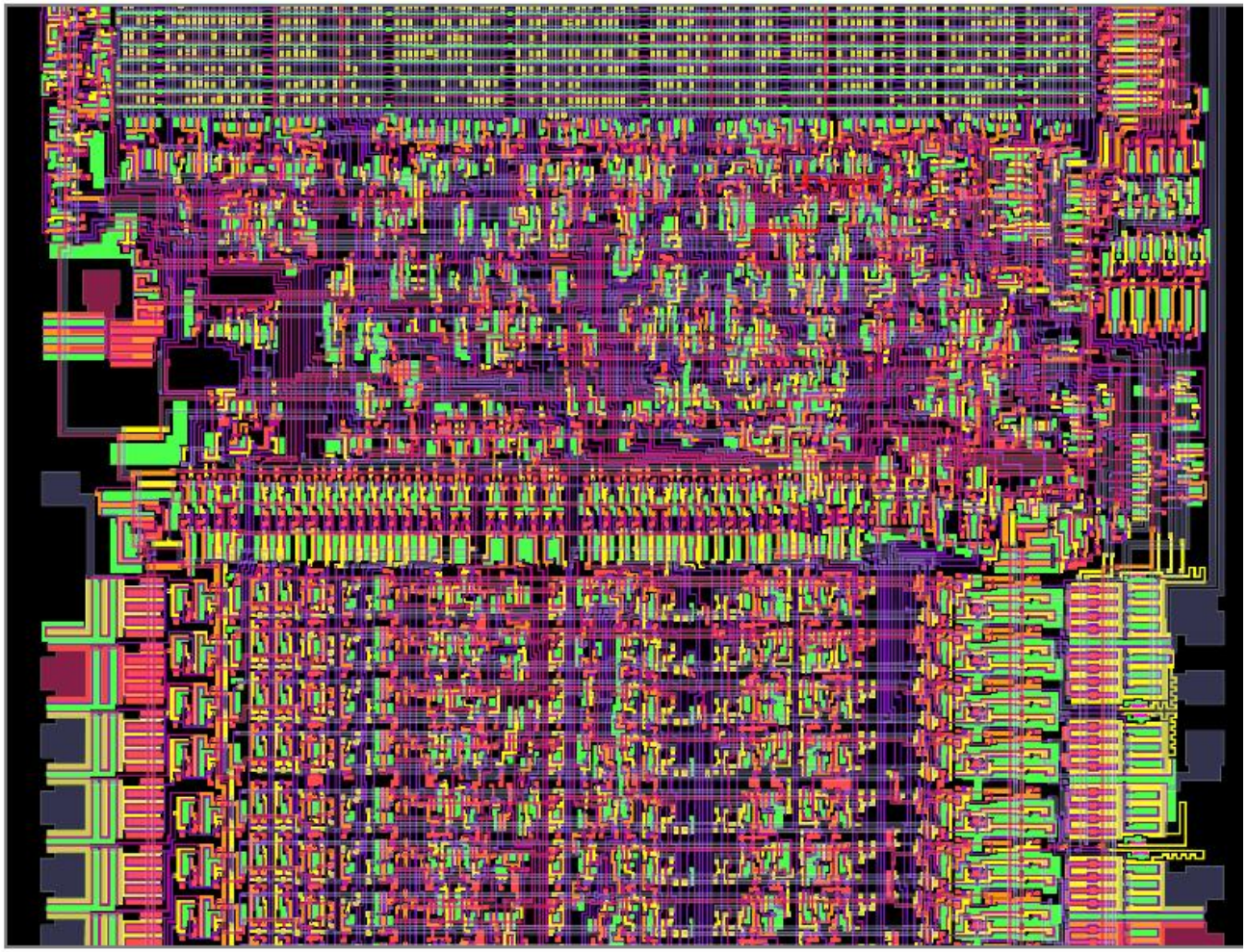
- Outras arquiteturas seguem os mesmos fundamentos.

Curiosidades



Processador

http://www.visual6502.org/JSSim/index.html



Introdução

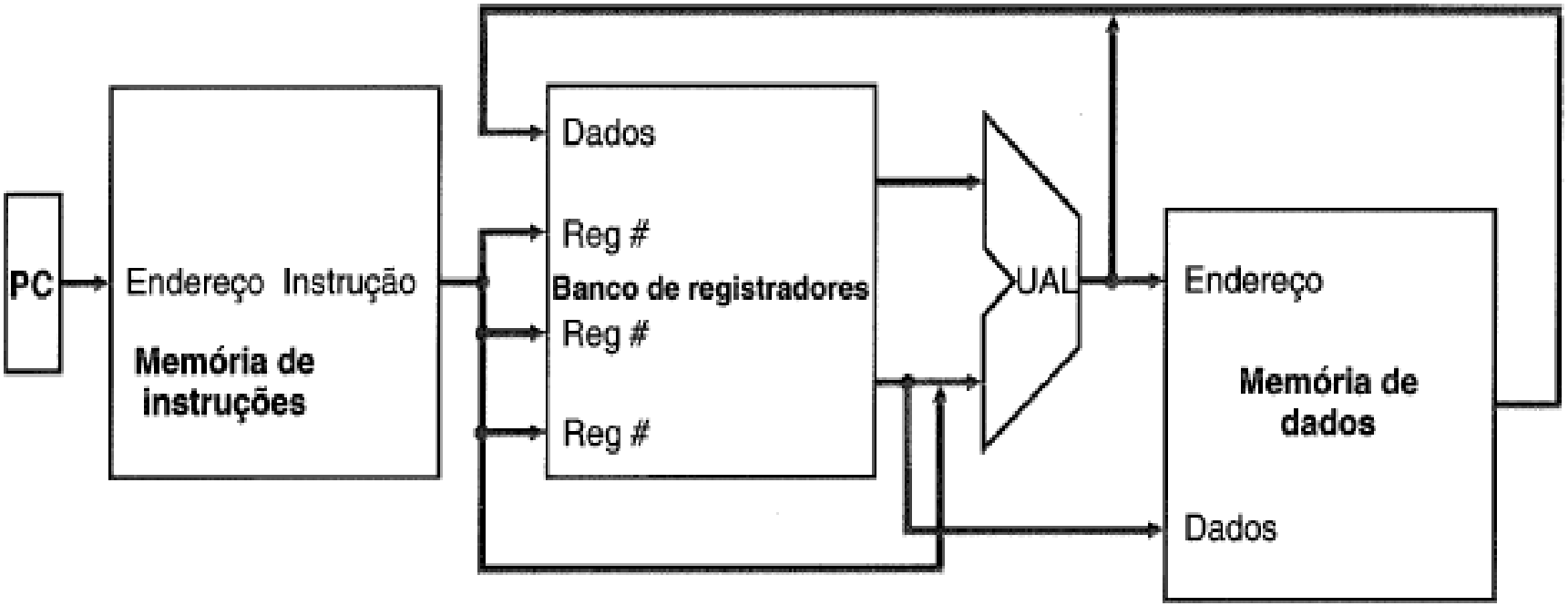
- Para qualquer instrução:
  - Passo 1: Enviar PC (Program counter) para a memória que contém o programa, e trazer para o caminho de dados a instrução armazenada nesse endereço da memória.
  - Passo 2: Ler um (ex. lw, sw) ou dois registradores (ex. Add, sub), usando os campos correspondentes da instrução lida.

Introdução

- Outras ações dependem da classe da instrução
  - Há muitas semelhanças entre elas.
  - Todas usam ULA após lidos os registradores.
    - As de acesso à mem. para cálculo do endereço
    - As aritméticas para a própria operação
    - As de desvios condicionais para a comparação
  - Similaridade facilita o projeto do HW!

- Depois da ULA, ações mudam um pouco
  - Acesso à mem endereça a mesma
  - Aritmética tem retorno em registradores
  - Desvios alteram o PC para a próxima instrução

Introdução


PC
Endereço
Instrução
Memória de instruções
Dados
Reg #
Banco de registradores
Reg #
Reg #
UAL
Endereço
Memória de dados
Dados

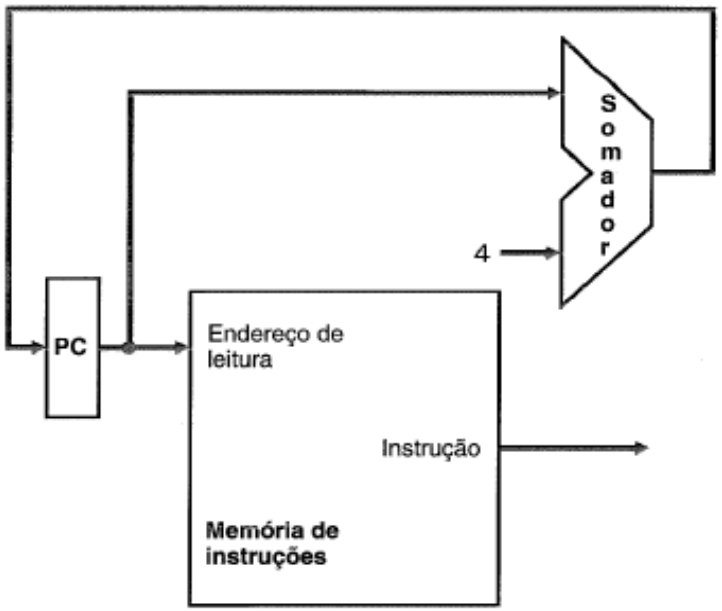
Considerações sobre Lógica e Clock

- Sistemas digitais são compostos por elementos:
  - Combinacionais, cujas saídas só dependem dos sinais de entrada
  - De estado, que têm capacidade de armazenar estado
- Clocks servem para determinar o momento de atualização de estado.
- Política de temporização para clocks é necessária para que circuitos funcionem com precisão.
- O processador pode ser implementado em um único ciclo de clock ou em vários
  - Um ciclo é mais simples, mas mostrou-se mais lento.
  - Vários ciclos exige maior controle.

MIPS Uniciclo

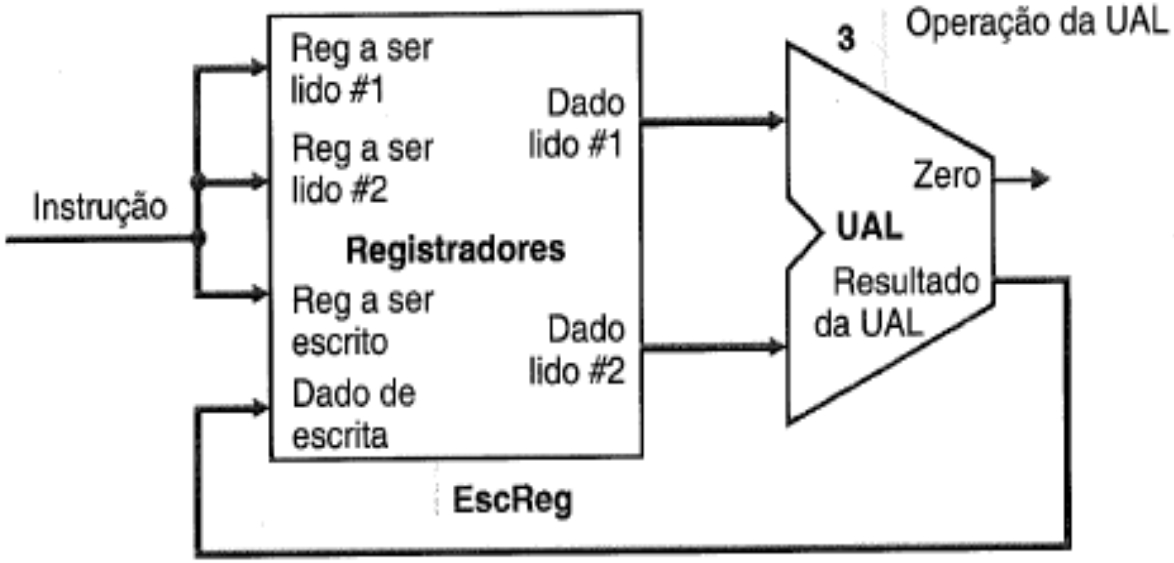
- Caminho de Dados
  - Interligação dos componentes básicos

- Busca de instruções necessita de:
  - Memória, onde instruções são armazenadas
  - PC, registrador para manter estado da próxima instrução a ser executada
  - Somador, para inc. do PC (+4 p/ restrição de alinham.)



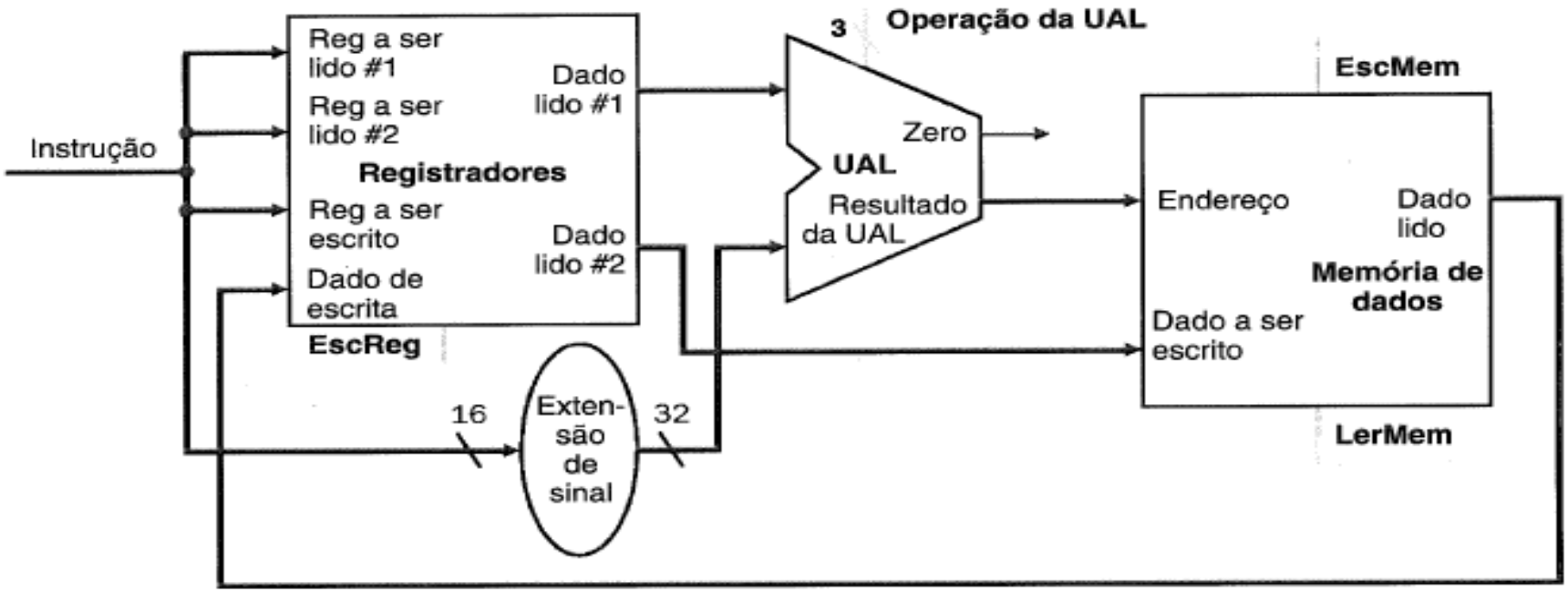
MIPS Uniciclo

- Instruções do tipo R
  - Lê registradores, operação ULA e escreve em registrador.
  - Registradores ficam em um banco de registradores
    - são acessados pelo seu número
    - Todo reg. De entrada gera seu dado na saída
  - Registrado a ser escrito precisa do número e do dado.
  - A ULA tem sua operação controlada por 3 bits (parte III)



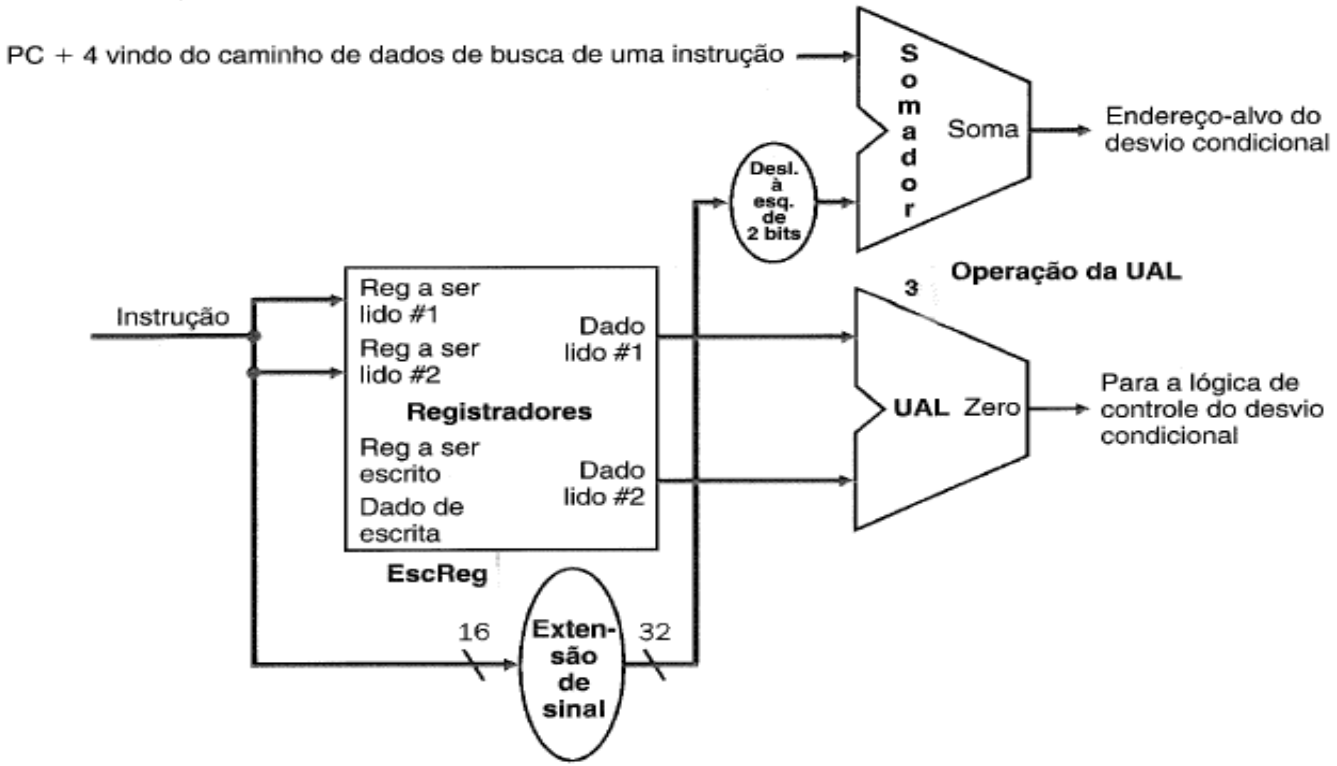
MIPS Uniciclo

- Operações de memória (ex. Lw $t1, desl($t2))
  - Calculam o end. De memória somando o conteúdo do registrador com o campo de deslocamento.
  - Se for store, valor a ser armazenado precisa ser lido do banco de registradores.
  - Se for load, o valor da memória deve ser escrito no banco de registradores.
  - O campo de desloc. precisa ser estendido de 16 para 32 bits.



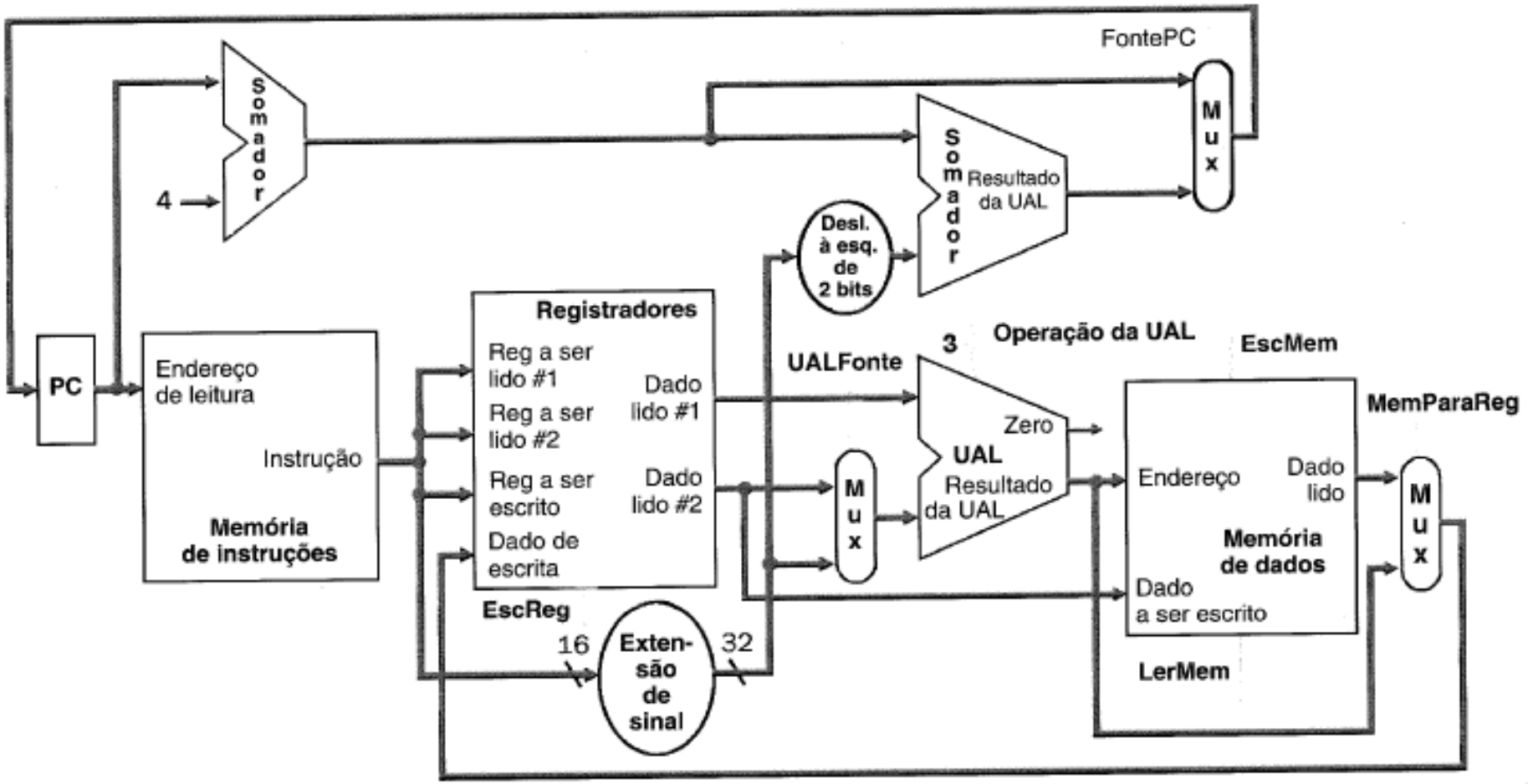
MIPS Uniciclo

- Instrução de branch condicional (beq)
  - Um desloc. (16 bits) como end. alvo deve ser estendido e somado ao PC
    - Como a ref é a próxima instrução, PC+4
    - Desloc. Deve ser deslocado 2 bits à esquerda (x4)
  - Próximo end. depende da comparação de 2 regs
    - Subtração na ALU com indicador de 0



MIPS Uniciclo

- Composição
  - Há reaproveitamento de componentes comuns, o que gera múltiplas entradas e saídas dos mesmos.
    - Seleção feita por multiplexadores (seletor de dados)



MIPS Uniciclo

- Controle da ULA
  - Já vimos anteriormente que a ULA tem entradas de controle que define a operação a ser executada.
  - São usadas diretamente pelas instruções aritméticas, e a operação é determinada pelo campo funct (função).
  - Operações de memória usam ULA para somar end. De memória.
  - Para o brach, ULA faz uma subtração (comparação se 0)

Entrada de controle da UAL	Função
000	AND
001	OR
010	Soma
110	Subtração
111	Set less than

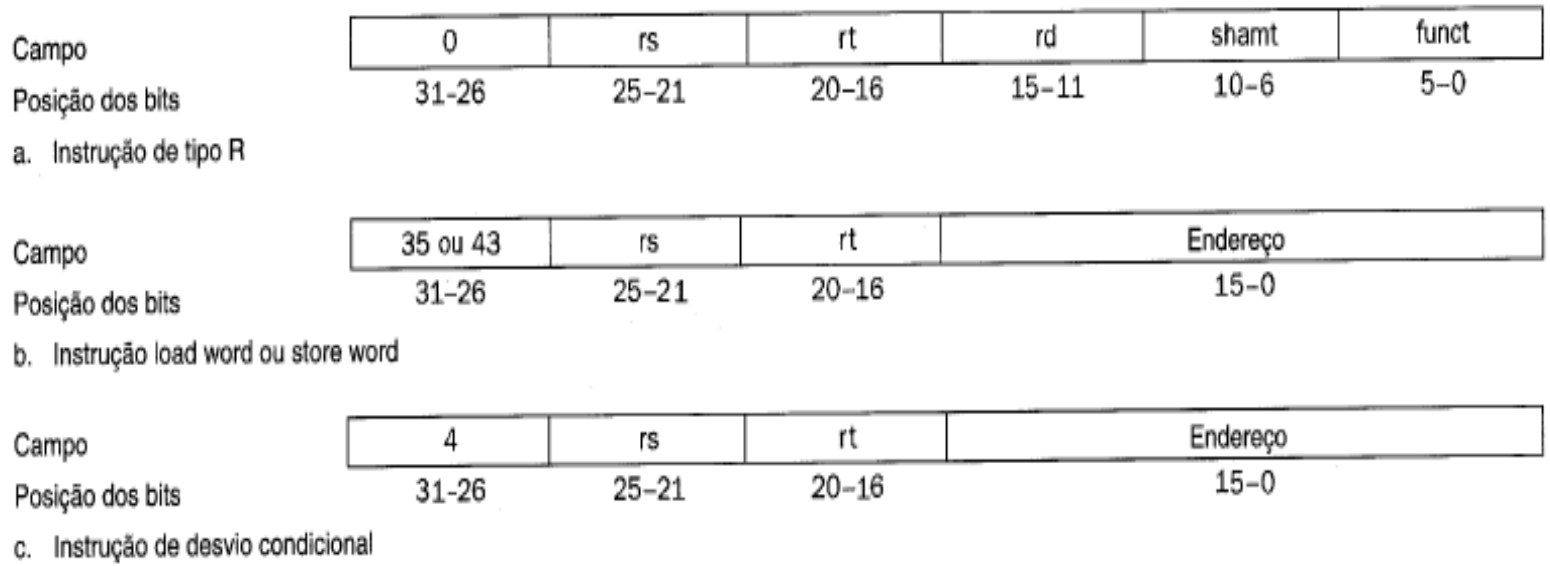
MIPS Uniciclo

- Controle ULA
  - Uma unidade de controle pode gerar esses 3 bits
  - Entradas UALOp e campo funct
  - O código da instrução determina UALOp

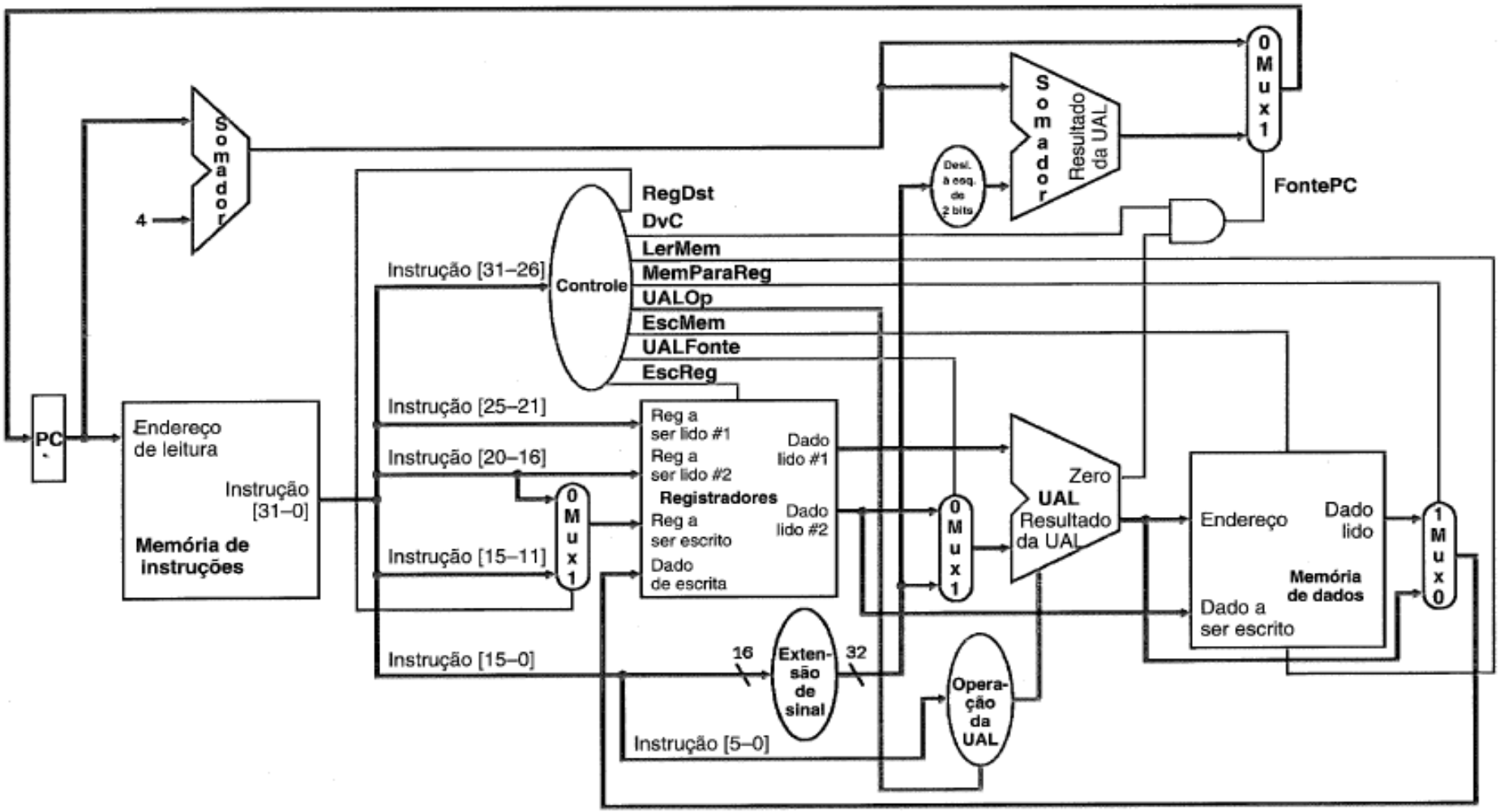
Código de operação da instrução	UALOp	Operação da instrução	Campo de Função	Operação desejada da UAL	Entrada de controle da UAL
LW	00	load word	XXXXXX	soma	010
SW	00	store word	XXXXXX	soma	010
Branch equal	01	branch equal	XXXXXX	subtração	110
Tipo R	10	add	100000	soma	010
Tipo R	10	subtract	100010	subtração	110
Tipo R	10	AND	100100	and	000
Tipo R	10	OR	100101	or	001
Tipo R	10	set less than	101010	set less than	111

MIPS Uniciclo

- Controle Principal
  - Extraídos dos campos da instrução
  - Revisão de formatos do MIPS:
    - Opcode nos bits 31-26
    - 2 Regs a serem lidos nas posições 25-21 e 20-16
    - Reg. Base p/ load e store sempre em 25-21
    - 16 bits de desloc. P/ branch, load e store em 15-0
    - Reg destino está de 15-11 se for instrução aritmética, senão, está em 20-16 no caso de load (uso de um MUX)



MIPS Controle



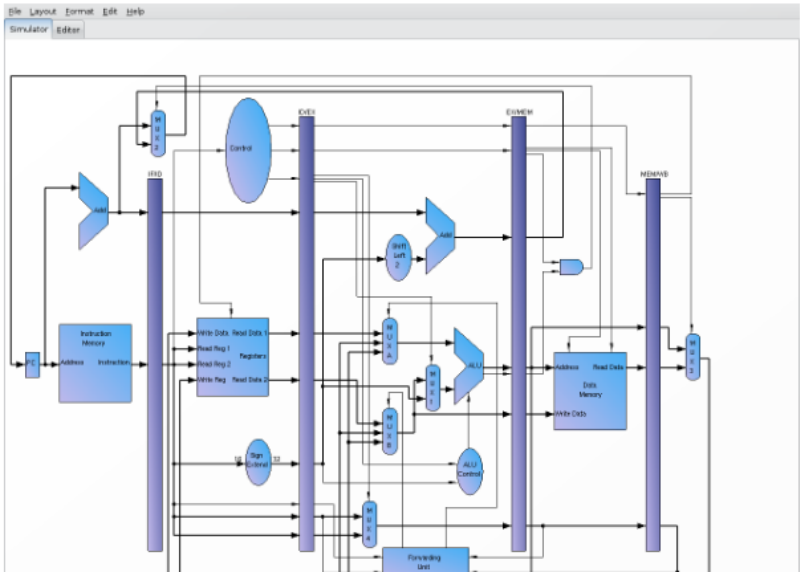
MIPS

http://mi.eng.cam.ac.uk/~ahg/MIPS-Datapath/

UNIVERSITY OF CAMBRIDGE DEPARTMENT OF ENGINEERING

MIPS-Datapath

MIPS-Datapath is a graphical MIPS CPU simulator. The program is intended to be used as a teaching aid for computer architecture courses involving MIPS. MIPS-Datapath simulates 10 different MIPS instructions (detailed in the user guide) with a graphical representation of the processor displaying how instructions are executed. The graphical user interface has an editor included allowing instructions to be written, and then parsed. These instructions are then fed into the simulator.



Load | Save | Parse

Instructions:

```
SUB $11, $9, $10
ADD $3, $1, $2
BEQ $9, 5, 4
SLT $6, $4, $5
AND $12, $9, $11
OR $56, $9, $4
SW $3, 6($4)
```

Messages:

```
There were 2 errors when parsing the instructions.
These errors are listed below and highlighted above.

'$' is required.
Register number was outside allowable bounds (0-31)
```

Praticando MIPS

```
int main()
{
    int n = 12;

    if (n%2 == 0)
       printf("Even\n");
    else
       printf("Odd\n");

    return 0;
}
```

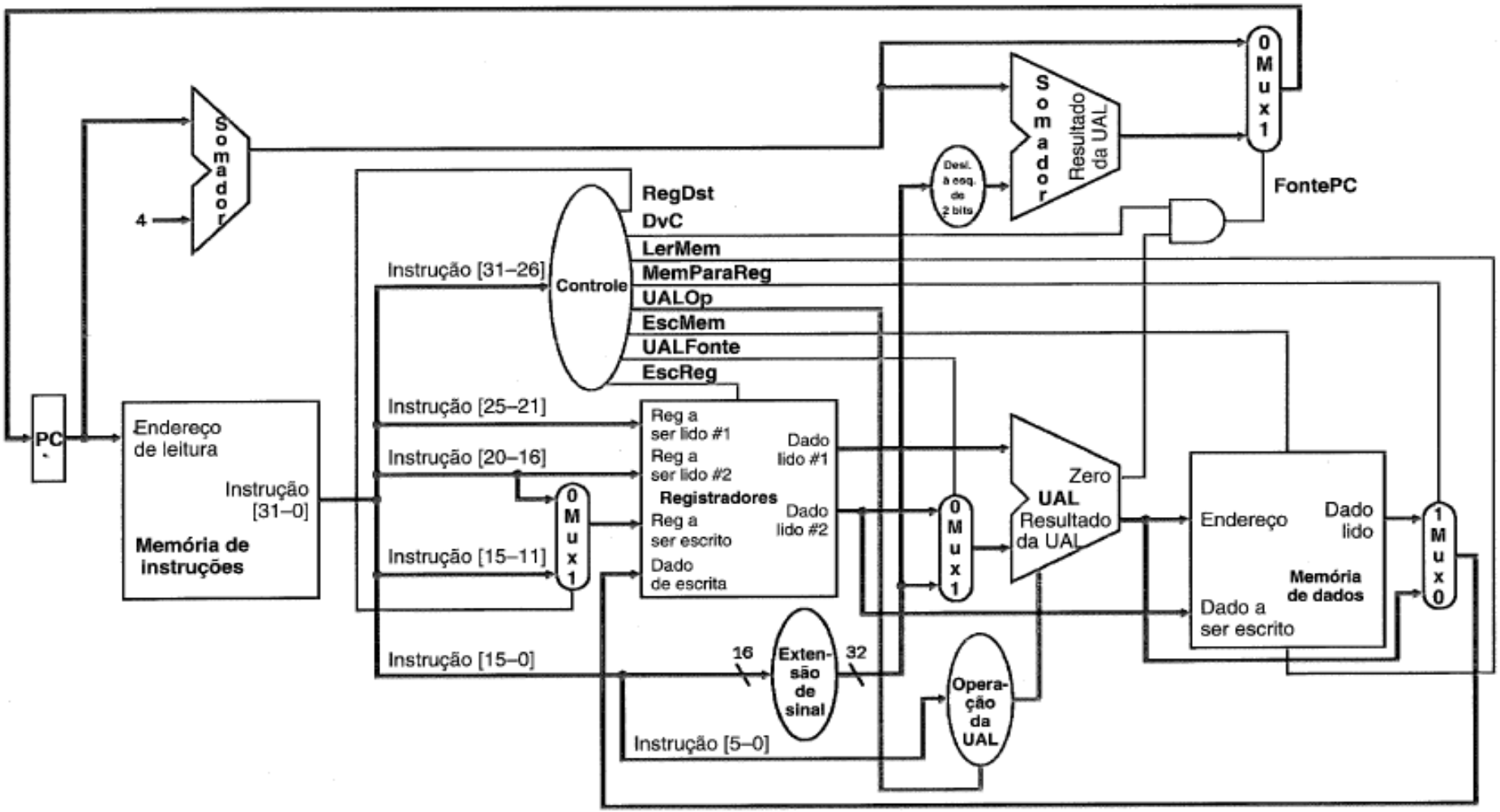
UNIVERSIDADE FEDERAL DE RORAIMA

Unidade de Controle

Baseado nas aulas do Prof. Dr. Mauricio Figueiredo - UFAM

Prof. Herbert Oliveira Rocha

MIPS Controle



MIPS Controle

Linhas de controle

Nome do sinal	Efeito quando ativo	Efeito quando inativo
RegDst	O número do registrador-destino onde será escrito o resultado da operação (Reg a ser Escrito) vem do campo rt (bits 20–16).	O número do registrador-destino onde será escrito o resultado da operação (Reg a ser Escrito) vem do campo rt (bits 15–11).
EscReg	Nenhum	O registrador na entrada Reg a ser Escrito é escrito com o valor presente na entrada Dado de Escrita.
UALFonte	O segundo operando da UAL vem do segundo registrador do banco de registradores (Dado lido #2).	O segundo operando da UAL é o resultado da extensão de sinal dos 16 bits menos significativos da instrução.
FontePC	O PC é substituído pelo valor presente na saída do somador que calcula PC + 4.	O PC é substituído pelo valor presente na saída do somador que calcula o endereço-alvo do desvio condicional.
LerMem	Nenhum	O conteúdo da memória designado pela entrada de endereço é colocado na saída Dado Lido.
EscMem	Nenhum	O conteúdo da memória designado pela entrada de endereço é substituído pelo valor presente na entrada Dado a ser Escrito.
MemParaReg	O valor na entrada do registrador de escrita vem da UAL.	O valor na entrada do registrador de escrita vem da memória de dados.

MIPS Controle

Sinais de Controle – MIPS Uniciclo

- P/ determiná-los, basta opcode da instrução, exceto para FontePC
- FontePC é 1, se instrução de branch e saída Zero da ULA.

Instrução	RegDst	UALFonte	MemParaReg	EscReg	LerMem	EscMem	DvC	UALOp1	UALOp0
Formato R	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

MIPS Controle

Sinais de controle - Uniciclo

- Opcodes:

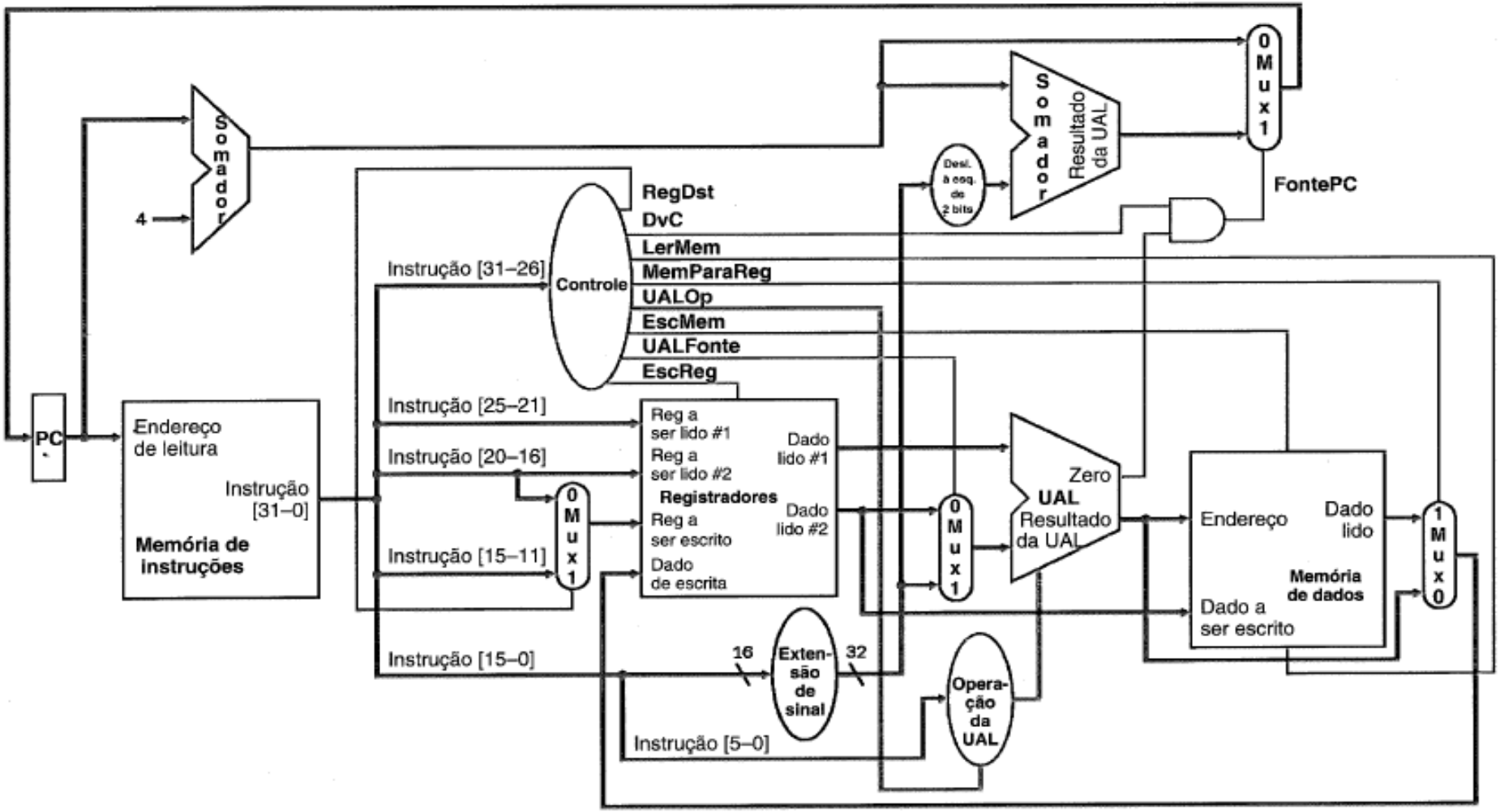
Nome	Opcode em decimal	Opcode em binário					
		Op5	Op4	Op3	Op2	Op1	Op0
Formato R	0_{dez}	0	0	0	0	0	0
lw	35_{dez}	1	0	0	0	1	1
sw	43_{dez}	1	0	1	0	1	1
beq	4_{dez}	0	0	0	1	0	0

- Determinação dos sinais:

Entrada ou saída	Nome do sinal	Formato R	lw	sw	beq
Entradas	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Saídas	RegDst	1	0	X	X
	UALFonte	0	1	1	0
	MemParaReg	0	1	X	X
	EscReg	1	1	0	0
	LerMem	0	1	0	0
	EscMem	0	0	1	0
	DvC	0	0	0	1
	UALOp1	1	0	0	0
	UALOp0	0	0	0	1

MIPS Controle - Uniciclo

- Visão Geral



MIPS Uniciclo

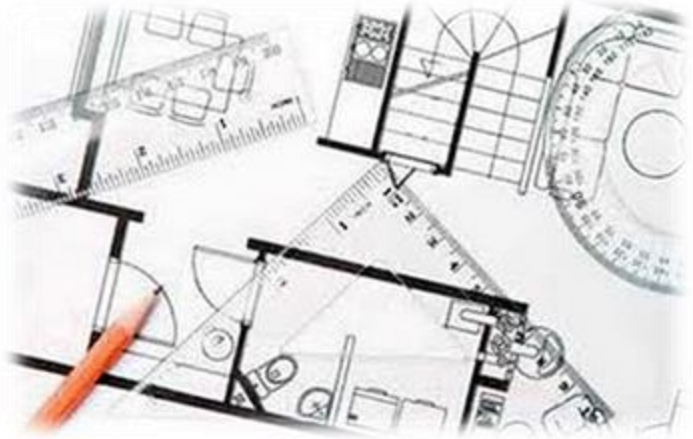
- Problemas
  - Todas as instruções deveriam funcionar em um único ciclo de clock (CPI = 1!)
  - O clock deverá ser ajustado para a instrução mais demorada
    - Ex. Load usa cinco unidades funcionais em série
    - Outras instruções poderiam ser executadas em ciclo menor
  - Ex. Se o tempo das unidade funcionais forem:
    - Memória: 2 ns
    - ULA: 2 ns
    - Banco de registradores: 1ns
    - Mux, controle e outros: Desprezíveis
    - Qual o clock para cada instrução?

Projeto Final

Projetando

Conjunto de instruções de 8bits

- Quantos registradores?
- Quais os formatos a serem suportados?
- Quais as operações a serem suportadas?
- Como será efetuado a separação de bits?



Projetando

Conjunto de instruções de 8bits

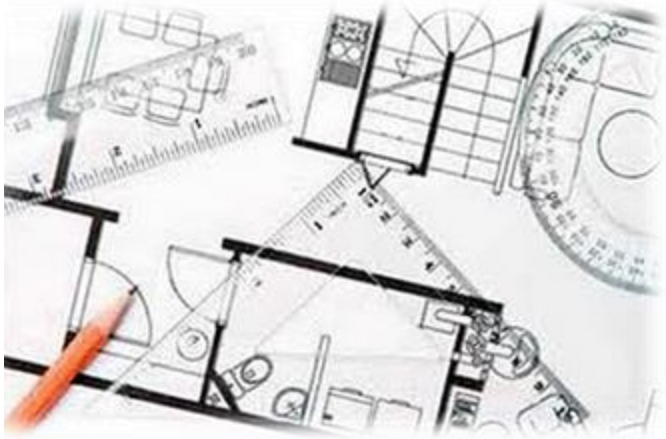
- Quantos registradores?
- Quais os formatos a serem suportados?
- Quais as operações a serem suportadas?
- Como será efetuado a separação de bits?

Formato para escrita de código na linguagem Quantum:

Tipo da Instrução	Reg1	Reg2

Formato para escrita em código binário:

4 bits	2 bits	2 bits
7-4	3-2	1-0
Opcode	Reg2	Reg1



Representação de Instruções

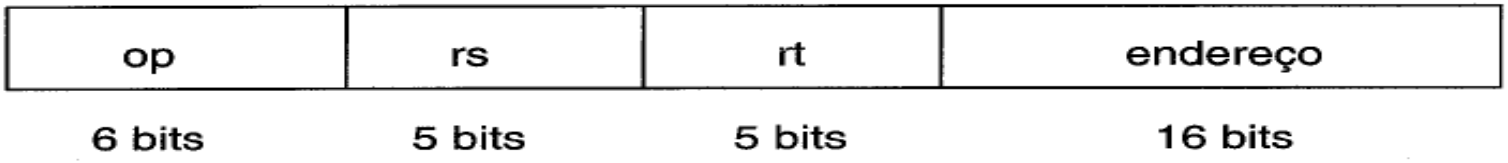
- Instruções tipo R (registradores como operandos)

A seguir, veja o significado de cada um dos campos das instruções do MIPS:

- *op*: a operação básica a ser realizada pela instrução, tradicionalmente chamada de *código de operação* (opcode)
- *rs*: o registrador contendo o primeiro operando-fonte
- *rt*: o registrador contendo o segundo operando-fonte
- *rd*: o registrador que guarda o resultado da operação, também conhecido como *registrador-destino*
- *shamt*: de quantidade de bits a serem deslocados. (Isto será explicado no Capítulo 4, quando estudarmos as instruções de deslocamento; até lá, este campo não será usado, e portanto conterá sempre zero.)
- *funct*: função. Este campo seleciona uma variação específica da operação apontada no campo op, sendo às vezes chamado de *código de função*.

Representação de Instruções

- Instruções tipo I (transf. De dados)
  - Campo endereço desloca o conteúdo de rs como base
  - Possibilita maior campo de endereçamento (mem. Grande)



- Instrução do tipo J (Jump)