MIPS Monociclo

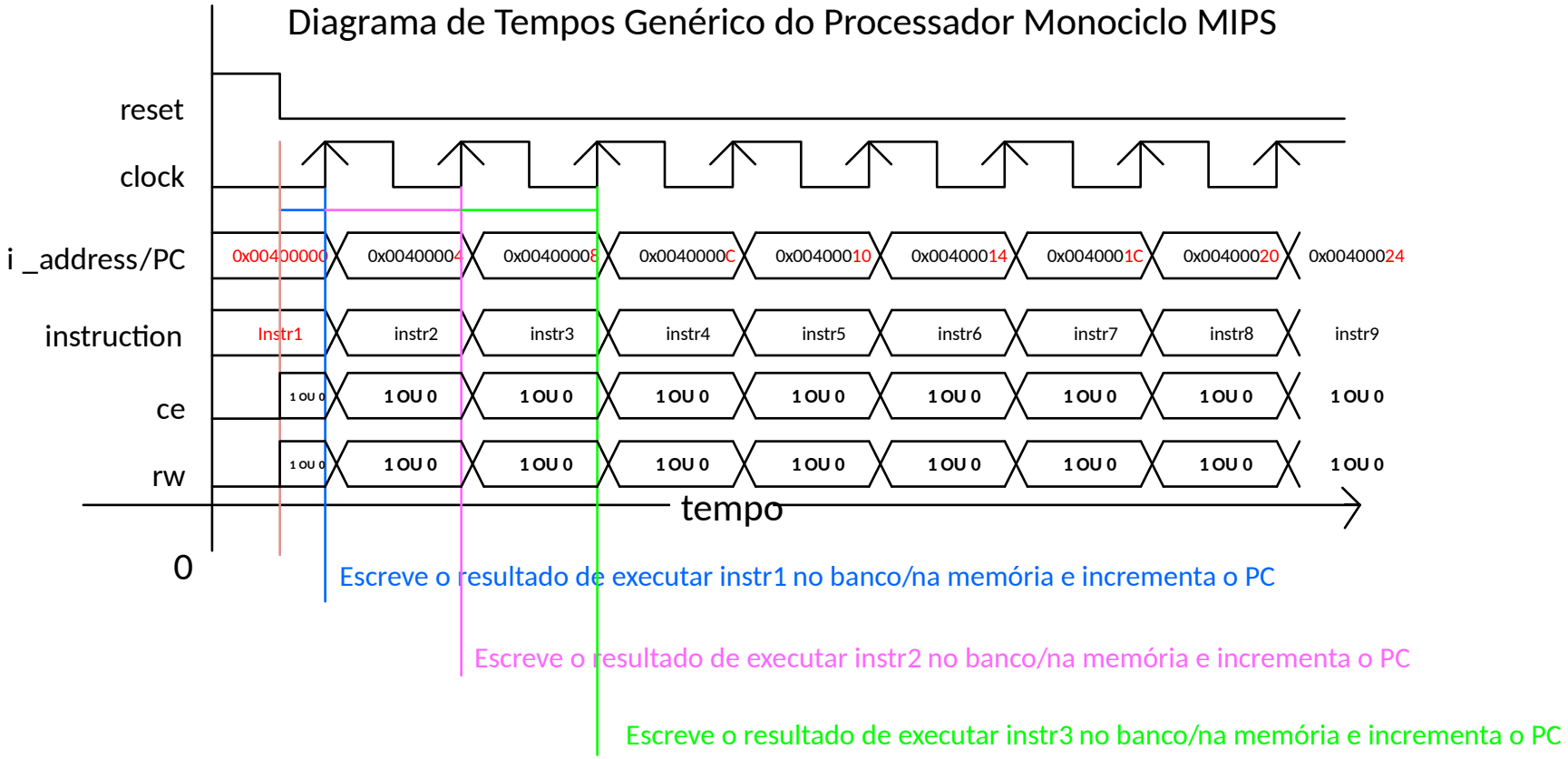
Especificação

Prof. Thiago Werlley

MIPS: Especificação Temporal

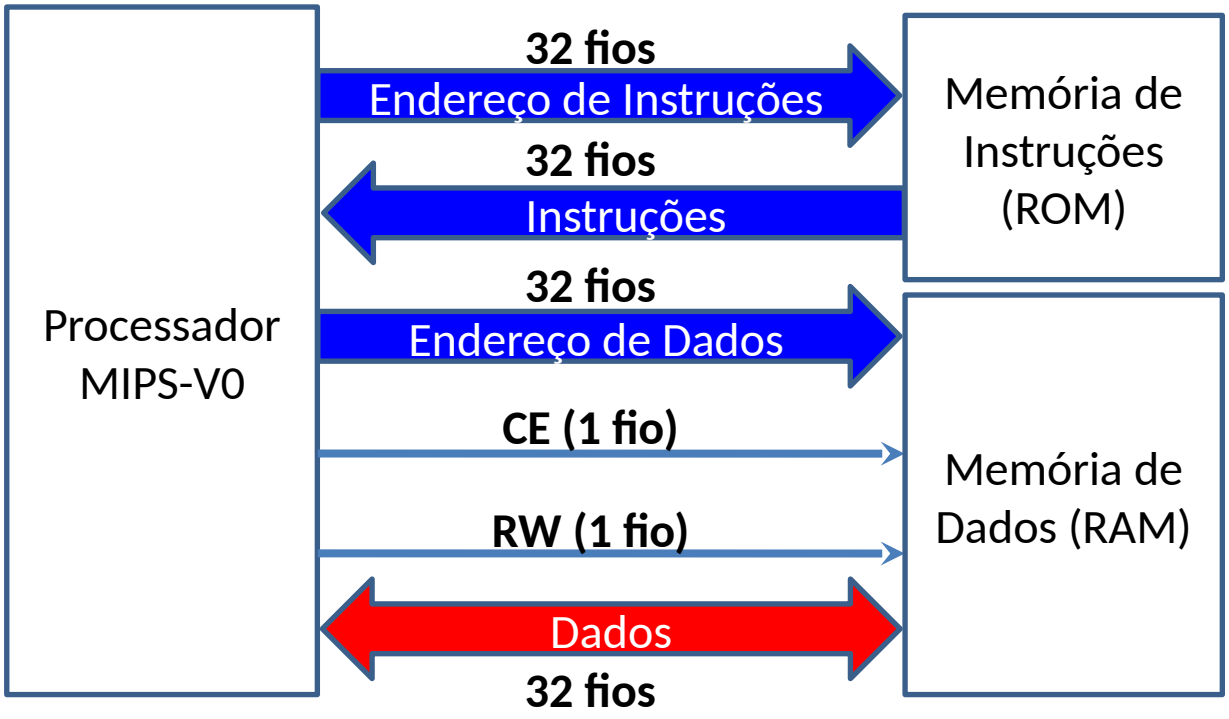
- Hardware síncrono (**`clock`**) com inicialização assíncrona (**`reset`**)
- Processador **monociclo**, ou seja, cada instrução gasta 1 ciclo de relógio (**`clock`**) para executar
- Sistema Digital sensível apenas à borda de subida do sinal de relógio
- Cada borda de subida do clock com inicialização desativada (Reset=0) - conclui execução de uma instrução e inicia a próxima
- Cada instrução executada - escrita do PC com seu valor anterior incrementado de 4 e escrita em algum registrador ou na memória (**nunca em ambos [registrador e memória] no mesmo ciclo!**)

MIPS: Operação Típica



MIPS: Especificação da Interface Processador-Memória – Organização Harvard

- Memórias endereçadas a byte – endereços de 32 bits
- Barramento de Dados único para a memória de dados – bidirecional, 32 bits
- Dois sinais de controle para acesso à memória de dados: CE, RW
- Na MIPS, cada leitura (e/ou escrita) faz acesso a 4 posições consecutivas de memória (a partir do endereço especificado em um Barramento de Endereços)



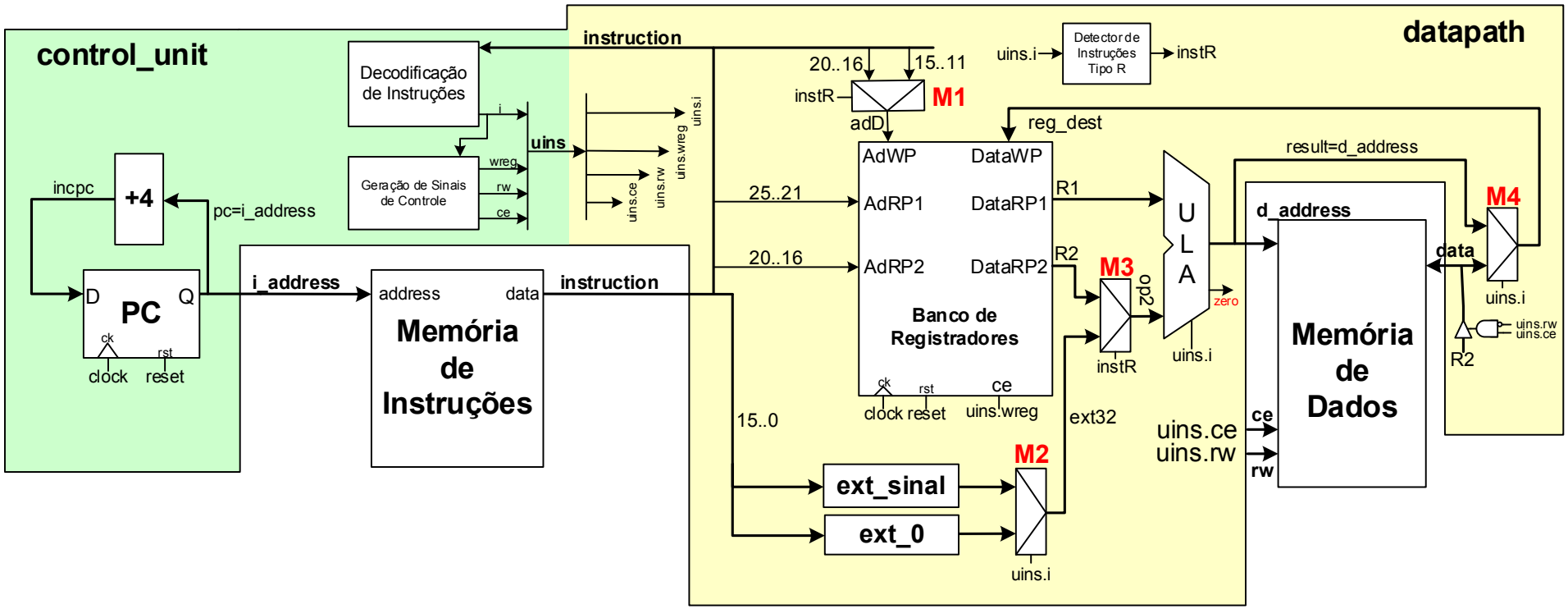
Especificação de ações associadas aos sinais de controle de acesso à Memória de Dados

CE	RW	Ação
0	-	Nenhuma
1	0	Escrita na Memória
1	1	Leitura da Memória

MIPS: Uma Proposta de Organização

- Processador - dividido em duas partes principais
  - Bloco de Dados - recebe, armazena dados e transforma-os
  - Bloco de Controle - comanda processo de execução de instruções
- O Bloco de Dados - Formado por
  - Banco de registradores (32 registradores de 32 bits)
  - ULA - 2 entradas de 32 bits, 1 saída de 32 bits, uma entrada de controle, 6 operações: +, -, AND, OR, XOR, NOR
  - Alguns blocos auxiliares (muxes, extensores de sinal, etc.)
- Bloco de Controle - possui três partes
  - Registrador PC, incrementável de 4 em 4
  - Decodificador de Instruções
  - Gerador de (outros) sinais de controle

Uma Proposta de Organização para a MIPS



- Organização Harvard - Memória de Instruções e Dados Separadas (externas à CPU)
- Região com fundo amarelo: Bloco de dados (nome em VHDL - datapath)
- Região com fundo verde: Bloco de controle (nome em VHDL - control_unit)
- Observar a interface com o resto do Universo (memórias)
- Memórias não são parte do processador (***testbench*** simula elas...)

MIPS: A Hierarquia da Organização

1. VHDL Auxiliar - Um *package* de definições
2. Cada bloco mencionado abaixo: deve ser desenvolvido em VHDL
   a) Registrador genérico - usado para o PC e no banco (**32 vezes**)
   b) ULA
   c) Banco de registradores - **32 registradores + controle de acesso**
   d) Bloco de Controle = registrador PC + código VHDL para incrementar PC, decodificar instruções e gerar sinais de controle para o Bloco de Dados
   e) Bloco de Dados - Une: Banco de registradores + ULA + código VHDL e contém estruturas auxiliares
   f) Processador - Une: Bloco de Dados + Bloco de Controle
3. Total: 6 pares entidade/arquitetura VHDL (E/A) distintos

MIPS: A Hierarquia da Organização

✓VHDL Auxiliar - Um *package* de definições

2. Cada bloco mencionado abaixo: deve ser desenvolvido em VHDL
   a) Registrador básico - usado para o PC e para o banco (32x)
   b) ULA
   c) Banco de registradores = 32 registradores
   d) Bloco de Controle = registrador PC + código VHDL para incrementar PC, decodificar instruções e gerar sinais de controle para o Bloco de Dados
   e) Bloco de Dados = Banco de registradores + ULA + código VHDL com estruturas auxiliares
   f) Processador - Junção Bloco de Dados + Bloco de Controle
3. Total: 6 pares E/A distintos

MIPS: A Hierarquia da Organização

✓VHDL Auxiliar - Um *package* de definições

2. Cada bloco mencionado abaixo: deve ser desenvolvido em VHDL
   - ✓Registrador genérico - usado para o PC e para o banco (32x)
   - b) ULA
   - c) Banco de registradores = 32 registradores
   - d) Bloco de Controle = registrador PC + código VHDL para incrementar PC, decodificar instruções e gerar sinais de controle para o Bloco de Dados
   - e) Bloco de Dados = Banco de registradores + ULA + código VHDL com estruturas auxiliares
   - f) Processador - Junção Bloco de Dados + Bloco de Controle
3. Total: 6 pares E/A distintos

MIPS: A Hierarquia da Organização

✓VHDL Auxiliar - Um *package* de definições

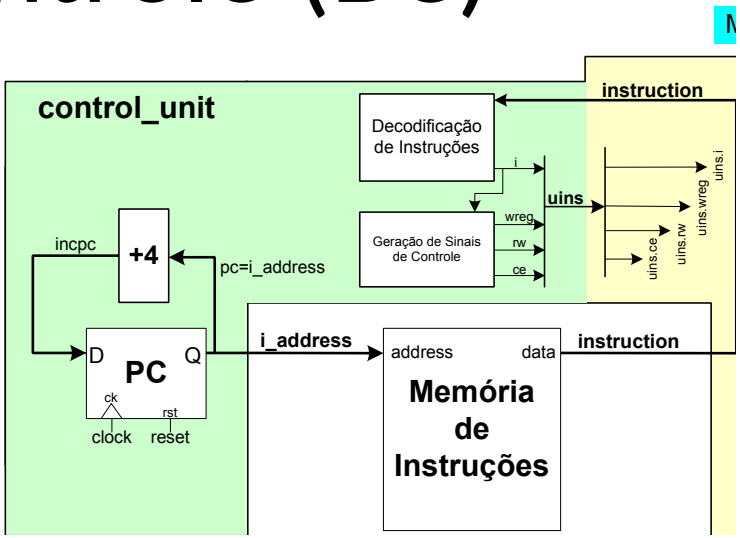
2. Cada bloco mencionado abaixo: deve ser desenvolvido em VHDL
   - ✓Registrador genérico - usado para o PC e para o banco (32x)
   - ✓ULA
   - c) Banco de registradores = 32 registradores
   - d) Bloco de Controle = registrador PC + código VHDL para incrementar PC, decodificar instruções e gerar sinais de controle para o Bloco de Dados
   - e) Bloco de Dados = Banco de registradores + ULA + código VHDL com estruturas auxiliares
   - f) Processador - Junção Bloco de Dados + Bloco de Controle
3. Total: 6 pares E/A distintos

MIPS: A Hierarquia da Organização

✓VHDL Auxiliar - Um *package* de definições

2. Cada bloco mencionado abaixo: deve ser desenvolvido em VHDL
   - ✓Registrador genérico - usado para o PC e para o banco (32x)
   - ✓ULA
   - ✓Banco de registradores = 32 registradores
   - d) Bloco de Controle = registrador PC + código VHDL para incrementar PC, decodificar instruções e gerar sinais de controle para o Bloco de Dados
   - e) Bloco de Dados = Banco de registradores + ULA + código VHDL com estruturas auxiliares
   - f) Processador - Junção Bloco de Dados + Bloco de Controle
3. Total: 6 pares E/A distintos

O Bloco de Controle (BC)



- Possui três partes
  - O decodificador de instruções (puramente combinacional)
  - A geração de sinais de controle para o Bloco de Dados, (puramente combinacional) quais sejam
    - Controles de leitura/escrita da/na memória (uins.ce e uins.rw)
    - Controle de escrita no banco de registradores (uins.wreg)
    - Código único para cada instrução (uins.i) – controla multiplexadores e operação da ULA
  - O Contador de Programa (PC) e seu incrementador

Formato R

Tamanho dos campos em bits:

6	5	5	5	5	6
31…26	25…21	20…16	15…11	10…6	5…0

ADDU Rd, Rs, Rt
0 Rs Rt Rd 0 0x21

SUBU Rd, Rs, Rt
0 Rs Rt Rd 0 0x23

AND Rd, Rs, Rt
0 Rs Rt Rd 0 0x24

OR Rd, Rs, Rt
0 Rs Rt Rd 0 0x25

XOR Rd, Rs, Rt
0 Rs Rt Rd 0 0x26

NOR Rd, Rs, Rt
0 Rs Rt Rd 0 0x27

Formato I

Tamanho dos campos em bits:

6	5	5	16
31…26	25…21	20…16	15…0

ORI Rt, Rs, imed
0x0d Rs Rt imed

LW Rt, offset (Rs)
0x23 Rs Rt offset

SW Rt, offset (Rs)
0x2b Rs Rt offset

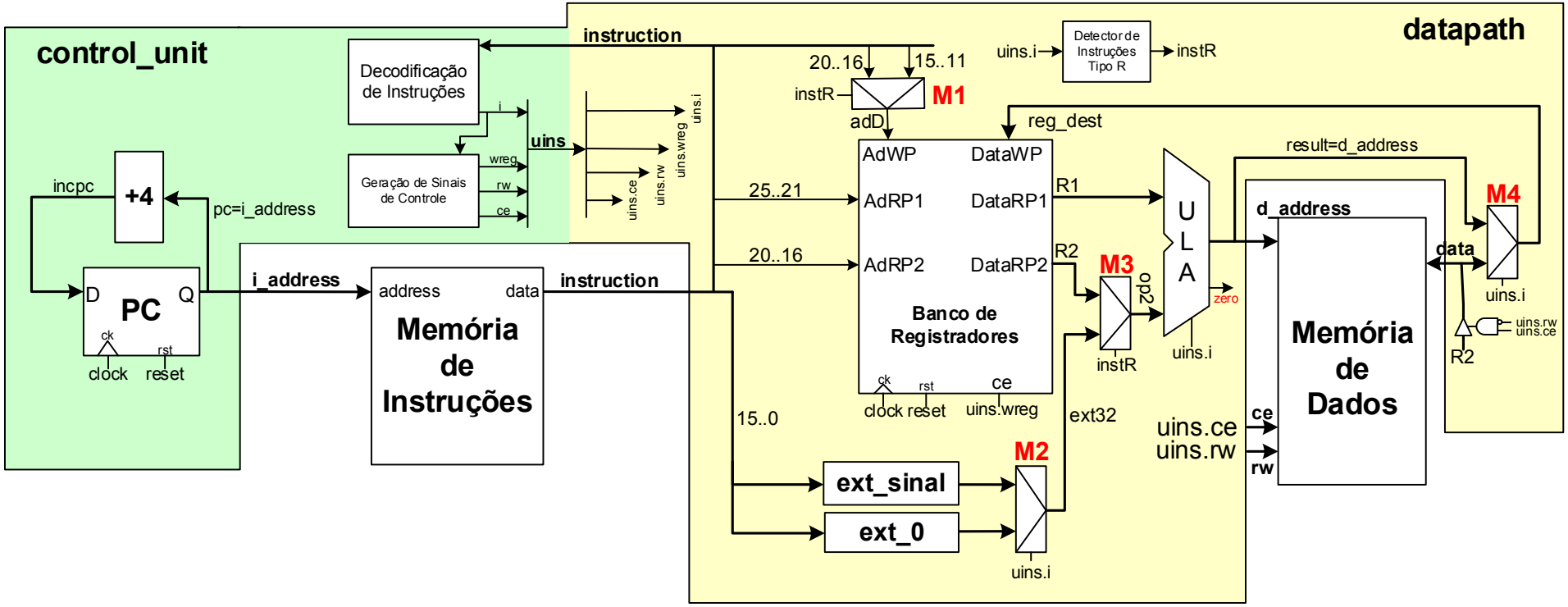
```
type inst_type is (ADDU, SUBU, AAND, OOR, XXOR, NNOR, LW, SW, ORI,
invalid_instruction)
```

MIPS: A Hierarquia da Organização

✓VHDL Auxiliar - Um *package* de definições

2. Cada bloco mencionado abaixo: deve ser desenvolvido em VHDL
   - ✓Registrador genérico - usado para o PC e para o banco (32x)
   - ✓ULA
   - ✓Banco de registradores = 32 registradores
   - ✓Bloco de Controle = registrador PC + código VHDL para incrementar PC, decodificar instruções e gerar sinais de controle para o Bloco de Dados
   - e) Bloco de Dados = Banco de registradores + ULA + código VHDL com estruturas auxiliares
   - f) O Processador = Junção Bloco de Dados + Bloco de Controle
3. Total: 6 pares E/A distintos

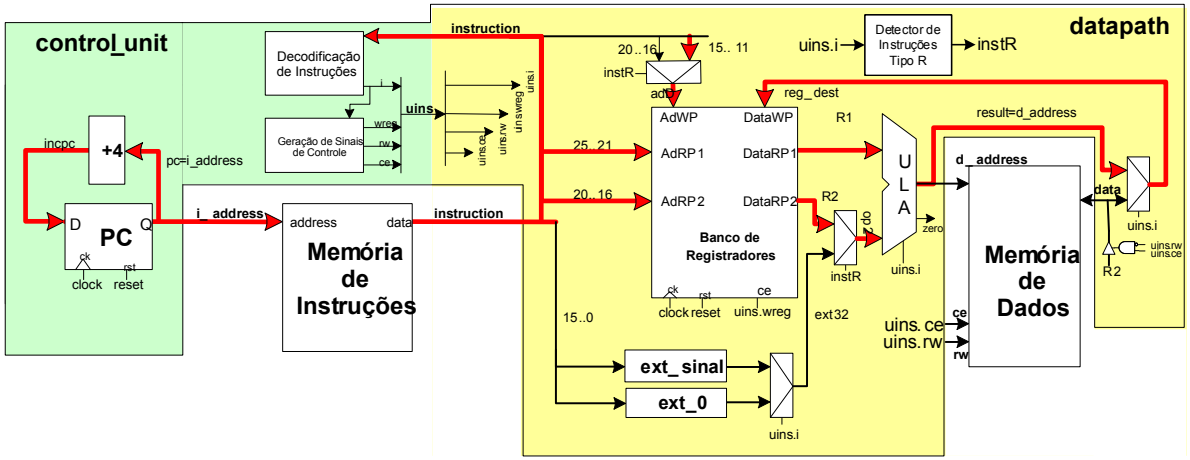
Diagrama de Blocos Completo do Processador (com parte do TB - memórias)



- Região de fundo amarelo: Bloco de dados (datapath)
- Região de fundo verde: Bloco de controle (control_unit)

O Bloco de Dados (BD)

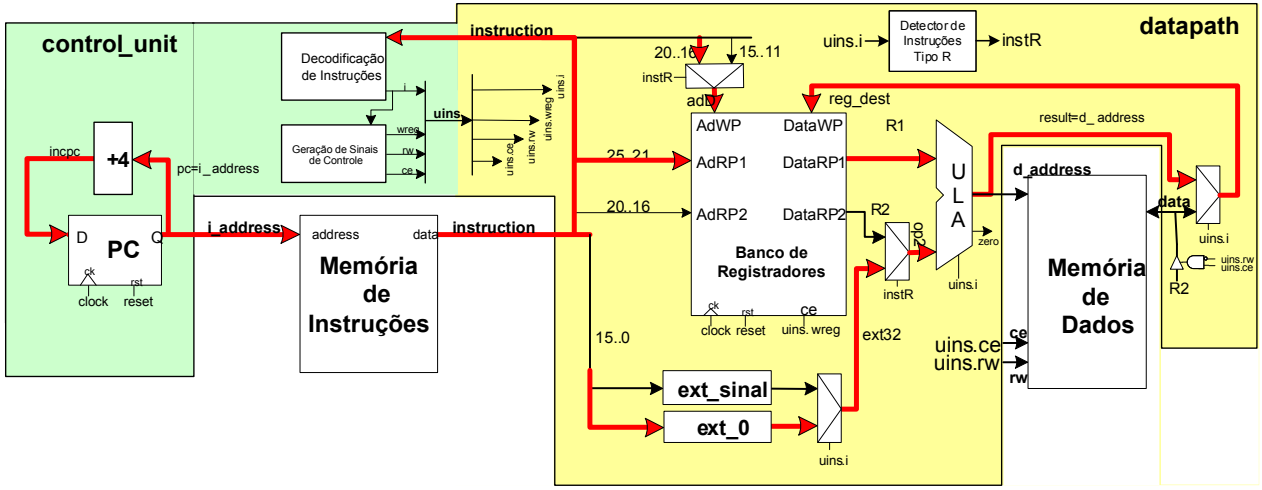
- Cada código objeto de uma instrução define seus operandos
- **O fluxo de dados das Instruções tipo R**



- Processo de controle dos quatro multiplexadores
  - `instR <= '1' when uins.i=ADDU or uins.i=SUBU or uins.i=AAND or uins.i=OOR or uins.i=XXOR or uins.i=NNOR else'0'; -- sinal auxiliar que define quando instrução é tipo R`
  - `adD <= instruction(15 downto 11) when instR='1' else instruction(20 downto 16); -- Mux: gera endereço de escrita no banco`
  - `op2 <= R2 when instR='1' else ext32; -- Mux: gera entrada inferior da ULA`
  - `reg_dest <= data when uins.i=LW else result; -- Mux: gera entrada da porta de escrita do Banco de Registradores`

O Bloco de Dados (BD)

- **O fluxo de dados para a Instrução ORI**

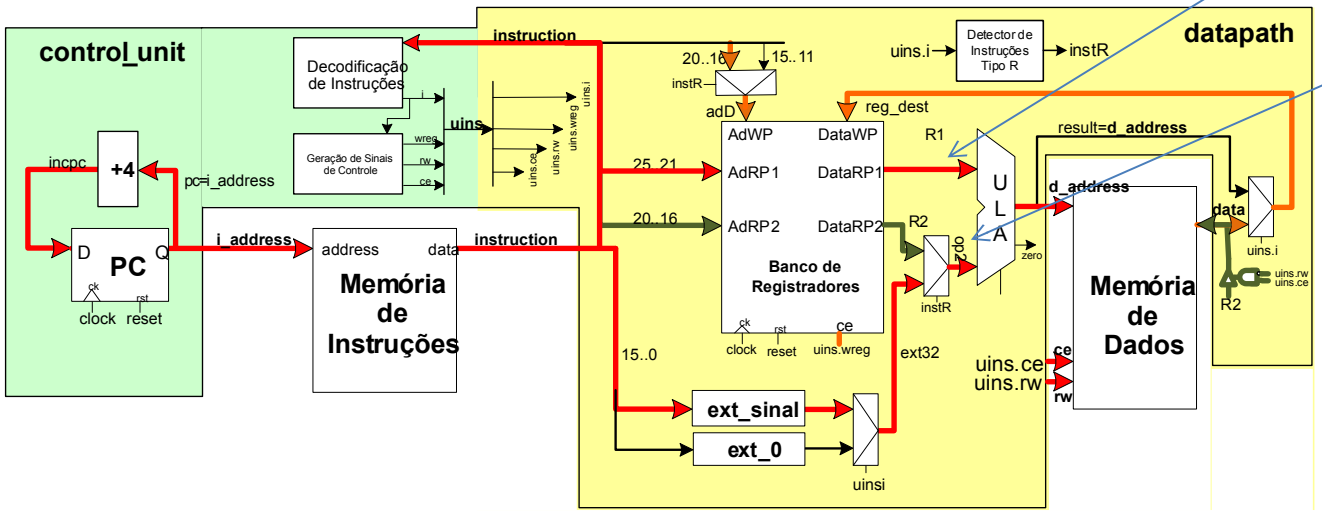


- 4 multiplexadores
  - `adD <= instruction(15 downto 11) when instR='1' else instruction(20 downto 16);`
  - `op2 <= R2 when instR='1' else ext32;  -- Mux: gera segunda entrada da ULA`
  - `ext32 <=x"FFFF" & instruction(15 downto 0) when (instruction(15)='1' and (uins.i=LW or uins.i=SW)) else x"0000" & instruction(15 downto 0); -- extensão de zero!`
  - `reg_dest <= data when uins.i=LW else result; -- Mux: gera entrada da porta de escrita do Banco de Reg`

O Bloco de Dados (BD)

- **O fluxo de dados para as Instruções LW / SW**

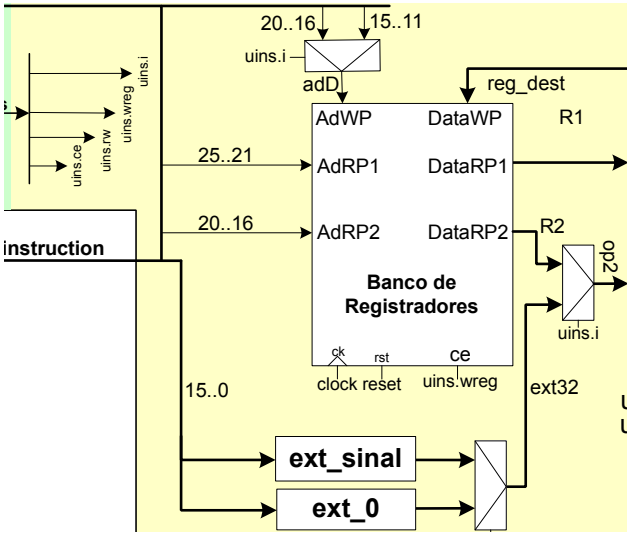
ENDEREÇO É A SOMA DO REG BASE + DESLOCAMENTO



Vermelho: LW/SW
Laranja: só LW
verdes: só SW

- 4 multiplexadores:
  - `adD <= instruction(15 downto 11) when instR='1' else instruction(20 downto 16) ;`
  - `op2 <= R2 when instR='1' else ext32; -- Mux: gera segunda entrada da ULA`
  - `ext32 <= x"FFFF" & instruction(15 downto 0) when (instruction(15)='1' and (uins.i=LW or uins.i=SW)) else x"0000" & instruction(15 downto 0);`
  - `reg_dest <= data when uins.i=LW else result; -- Mux: gera entrada da porta de escrita do Banco de Registradores`

O Código VHDL do BD

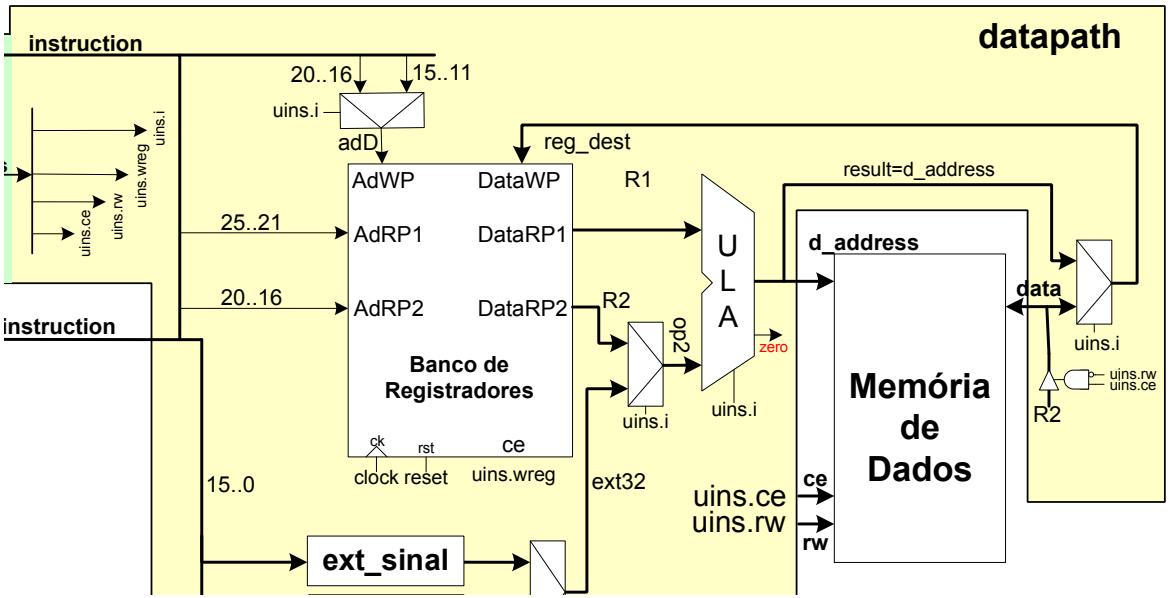



```
--==== hardware do banco de registradores e extensao de sinal ou de 0 ==
   adD <= instruction(15 downto 11) when instR='1'  else
          instruction(20 downto 16);

   REGS: entity work.reg_bank port map
        (ck=>ck, rst=>rst, ce=>uins.wreg, AdRP1=>instruction(25 downto 21),
        AdRP2=>instruction(20 downto 16), AdWP=>adD,
        DataWP=>reg_dest, DataRP1=>R1, DataRP2=>R2);

   -- Extensao de 0 ou extensao de signal
   ext32 <= x"FFFF" & instruction(15 downto 0) when (instruction(15)='1'
     and (uins.i=LW or uins.i=SW)) else
     -- LW e SW usam extensao de sinal, ORI usa extensao de 0
     x"0000" & instruction(15 downto 0);
     -- outras instrucoes nao usam esta informacao,
     -- logo, qualquer coisa serve, extensao de 0 ou de sinal
```

O Código VHDL do BD




```
--==========  hardware da ALU e em volta dela
   op2 <= R2 when instR='1' else ext32;

   inst_alu: entity work.alu port map (op1=>R1, op2=>op2,
         outalu=>result, zero=>zero, op_alu=>uins.i);

-- operacao com a memoria de dados
   d_address <= result;

   data <= R2 when uins.ce='1' and uins.rw='0' else (others=>'Z');

   reg_dest <=  data when uins.i=LW else result;

end datapath;
```

MIPS: A Hierarquia da Organização

✓VHDL Auxiliar - Um *package* de definições

2. Cada bloco mencionado abaixo: deve ser desenvolvido em VHDL
   - ✓Registrador genérico - usado para o PC e para o banco (32x)
   - ✓ULA
   - ✓Banco de registradores = 32 registradores
   - ✓Bloco de Controle = registrador PC + código VHDL para incrementar PC, decodificar instruções e gerar sinais de controle para o Bloco de Dados
   - ✓Bloco de Dados = Banco de registradores + ULA + código VHDL com estruturas auxiliares

   f) O Processador - Junção Bloco de Dados + Bloco de Controle

3. Total: 6 pares E/A distintos