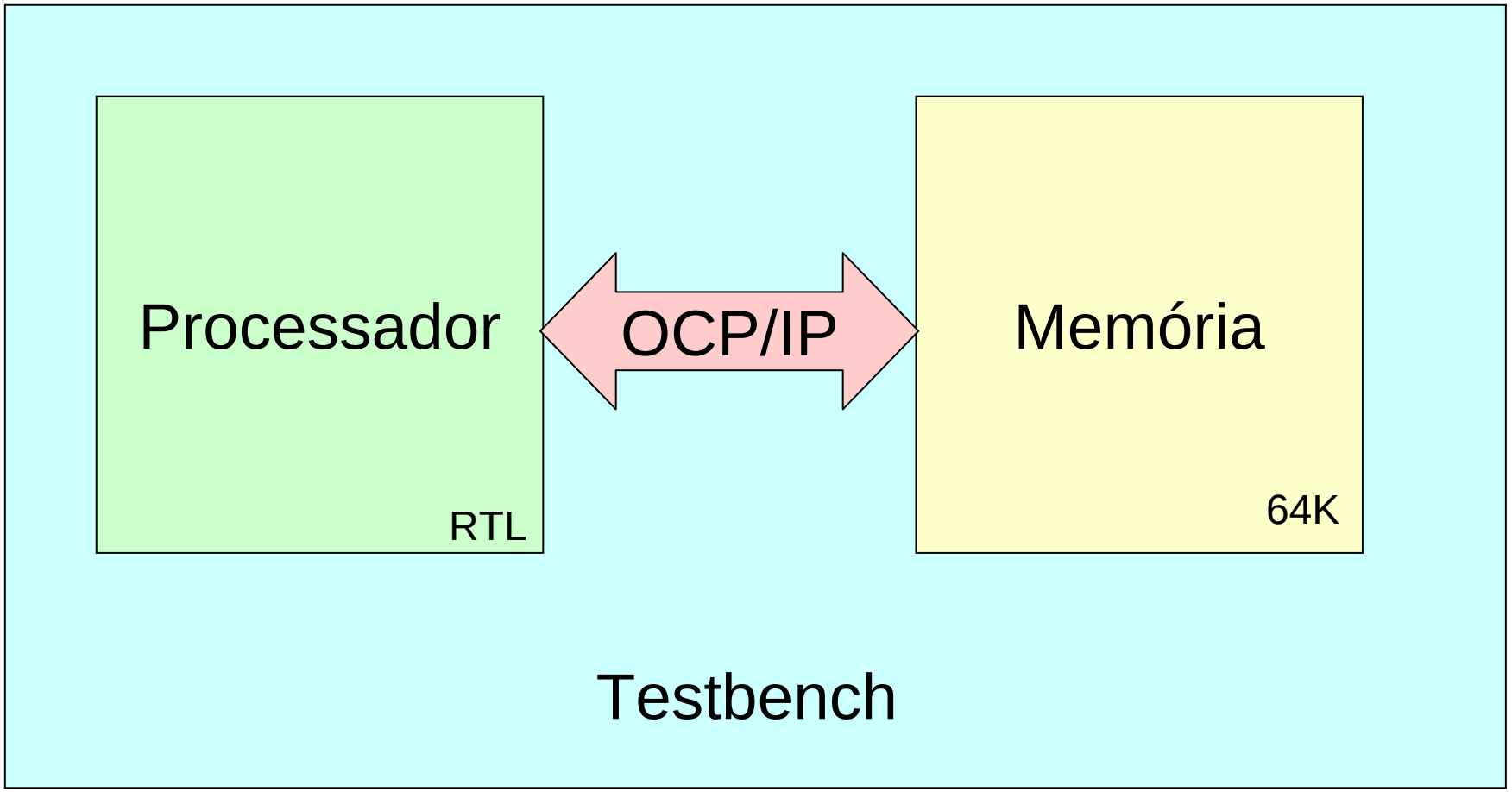
Trabalho 2
Processador Simples

MO801/MC912

Objetivos

- Implementar um processador com um pequeno conjunto de instruções
- Implementar um testbench para a especificação do processador
- Implementar uma memória externa e usá-la com o processador

Diagrama



Sinais OCP/IP

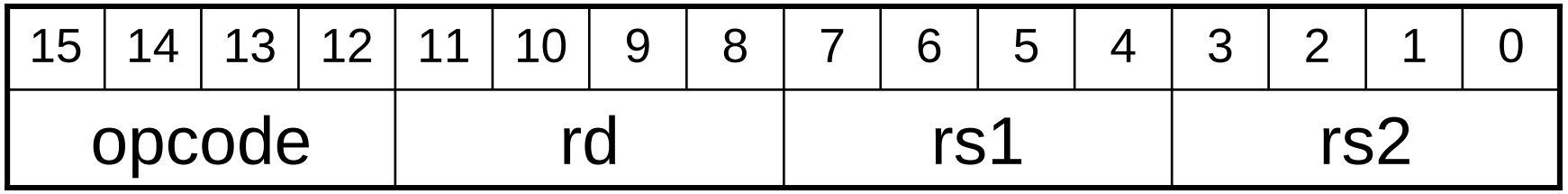
- Somente os seguintes sinais OCP/IP devem ser utilizados
  - Clk, Resetn, MCmd, MAddr, MData, SCmdAccept, SResp, SData
- Endereços e dados de 16 bits
- Não utilizar transferências com pipeline
- Utilizar os nomes dos sinais OCP como portas da entidade principal

O Processador

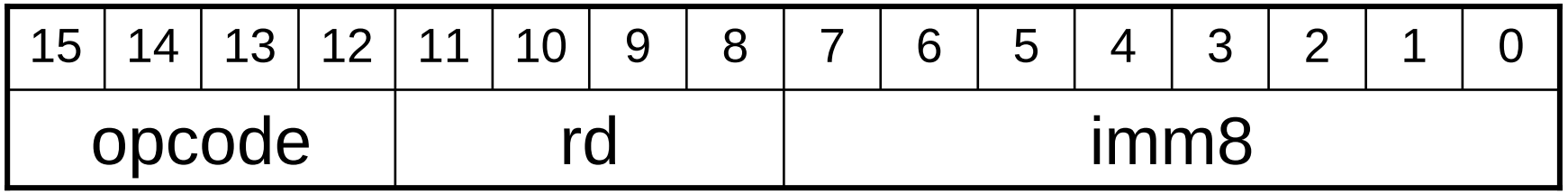
- Implementação multiciclo
  - Não vale implementação em pipeline
  - Não vale um único ciclo para todas as instruções
- 16 instruções de 16 bits
- 16 registradores de 16 bits
  - Leitura do registrador 0 resulta sempre no valor 0
- 3 formatos de instruções
- Cada instrução pode durar o tempo que for necessário
- Começa a executar a partir do endereço 0

Formato das Instruções

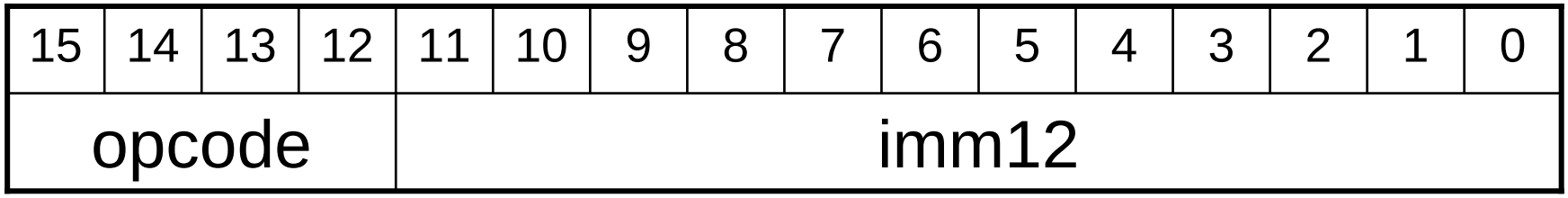
- Formato R



- Formato I



- Formato J



Instruções (opcode)

- add (0)
- sub (1)
- and (2)
- nor (3)
- slt (4)
- mult (5)
- beq (6)
- bne (7)
- call (8)
- jump (9)
- load (10)
- store (11)
- lui (12)
- li (13)
- shl (14)
- shr (15)

add

- Formato R
- Sintaxe
  - add rd, rs1, rs2
- Operação

  rd = rs1 + rs2

sub

- Formato R
- Sintaxe
  - sub rd, rs1, rs2
- Operação

  rd = rs1 - rs2

and

- Formato R
- Sintaxe
  - and rd, rs1, rs2
- Operação

  rd = rs1 and rs2
  - operação bit a bit

nor

- Formato R
- Sintaxe
  - nor rd, rs1, rs2
- Operação

  rd = rs1 nor rs2
  - operação bit a bit

slt

- Formato R
- Sintaxe
  - slt rd, rs1, rs2
- Operação

  rd = (rs1 < rs2) ? 1 : 0

- Serve para testar condições
- Use em conjunto com as instruções de salto para implementar ifs

mult

- Formato R
- Sintaxe
  - mult rd, rs1, rs2
- Operação

  rd = rs1 * rs2

beq

- Formato R
- Sintaxe
  - beq rd, rs1, rs2
- Operação

  if (rs1 == rs2)

    PC = rd

bne

- Formato R
- Sintaxe
  - bne rd, rs1, rs2
- Operação

  if (rs1 != rs2)

    PC = rd

call

- Formato R
- Sintaxe
  - call rd, rs1
- Operação
  - rd = PC + 2
  - PC = rs1

- Para chamar um procedimento

  call r15, r3
- Para retornar de um procedimento

  call r0, r15

jump

- Formato J
- Sintaxe
  - jump imm12
- Operação
  - PC[12 downto 1] = imm12

load

- Formato R
- Sintaxe
  - load rd, rs1
- Operação

  rd = MEM[rs1]

store

- Formato R
- Sintaxe
  - store rs1, rs2
- Operação

  MEM[rs1] = rs2

lui

- Formato I
- Sintaxe
  - lui rd, imm8
- Operação

  rd[15 downto 8] = imm8

li

- Formato I
- Sintaxe
  - li rd, imm8
- Operação

  rd[7 downto 0] = imm8

shl

- Formato R
- Sintaxe
  - shl rd, rs1, rs2
- Operação

  rd = rs1 << rs2
  - Obs.: deve ser usado o valor do campo rs2 e não o do registrador indicado por rs2

shr

- Formato R
- Sintaxe
  - shr rd, rs1, rs2
- Operação

  rd = rs1 >> rs2
  - Obs.: deve ser usado o valor do campo rs2 e não o do registrador indicado por rs2

Observações

- Não foram definidas convenções de registradores nem de passagem de parâmetros
- A memória faz parte do testbench
  - Não precisa ser RTL

Estrutura da entrega

- Arquivo no formato .tgz contendo um diretório nomeado pelo RA do aluno. Dentro dele:
  - Um diretório com a implementação
  - Um diretório com o testbench
  - Um script para compilar a implementação
  - Um script para compilar o testbench
  - Um script para executar o testbench