# Design de Computadores

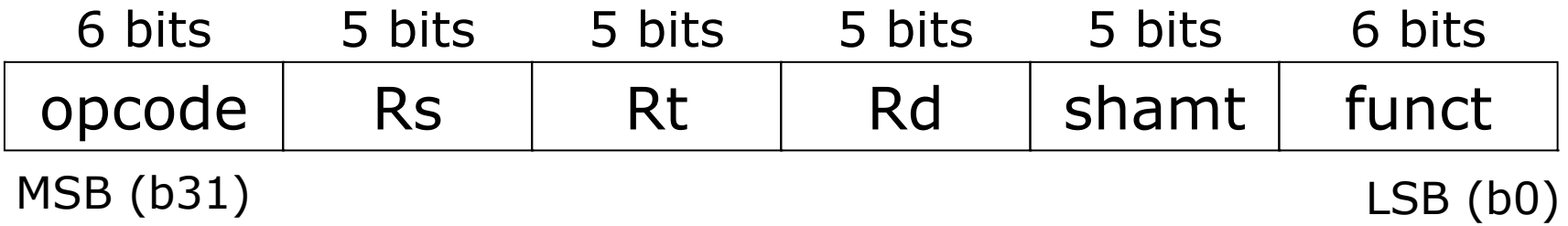
Aula 10

Insper

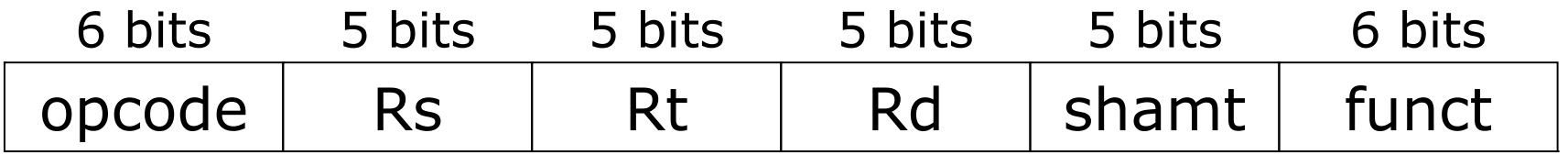
## Decomposição de Instruções
## Tipo R

- Tópicos:
  - Formato das instruções;
  - Codificação das instruções;
  - Análise do funcionamento das instruções;
  - Esboço de fluxo de dados que execute essas instruções;
  - Simulação manual do funcionamento do FD;
  - Implementação do FD em VHDL.

- Como visto anteriormente, o MIPS possui:
  - 3 formatos básicos de instruções:
    - Tipo R (registro);
    - Tipo I (imediato);
    - Tipo J (salto).
- Além disso, a codificação dessas instruções:
  - Utiliza uma palavra de 32 bits;
  - Que foi dividida em campos bem definidos.



- Esses campos podem ser agrupados:
  - Dependendo do tipo de instrução.
- O significado dos nomes dos campos:
  - **Opcode**: Contém o código da instrução a ser executada;
  - **Rs**: O número do registrador com o primeiro operando da instrução definida em op;
  - **Rt**: O número do registrador com o segundo operando da instrução definida em op;
  - **Rd**: O número do registrador de destino do resultado da instrução definida em op;
  - **shamt**: Total de deslocamento (shift amount).
  - **funct**: Seleciona uma variação específica da operação definida em opcode.



- Analisaremos o formato das instruções do tipo R:
  - Para determinar a estrutura de hardware necessária para implementar esse tipo de instrução.
- Todas instruções do tipo R:
  - Utilizam 3 registradores;
  - O campo opcode é sempre igual a zero;
  - O campo “funct” define a função a ser realizada.
- A função dos registradores:
  - Rs e Rt contém os argumentos da operação;
  - Rd é o destino do resultado da operação.

- Para a soma mostrada abaixo:

  add $t0, $s1, $s2

- Qual seria a sua codificação?
  - Dica:
    - Consulte valor de “funct” e o endereço dos registradores no **greencard**.

| 6 bits | 5 bits | 5 bits | 5 bits | 5 bits | 6 bits |
|---|---|---|---|---|---|
| 31~26 | 25 ~21 | 20 ~16 | 15 ~11 | 10 ~6 | 5 ~0 |
| opcode | Rs | Rt | Rd | shamt | funct |
| 0x00 | | | | | |

- E para a subtração mostrada abaixo:

sub $a0, $sp, $s2

- Qual seria a sua codificação?
  - Dica: consulte o **greencard**.

| 6 bits | 5 bits | 5 bits | 5 bits | 5 bits | 6 bits |
|---|---|---|---|---|---|
| 31~26 | 25 ~21 | 20 ~16 | 15 ~11 | 10 ~6 | 5 ~0 |
| opcode | Rs | Rt | Rd | shamt | funct |
| 0x00 | | | | | |

# Blocos Construtivos

- Lembrando dos blocos construtivos que conhecemos:
  - Registrador de uso geral (n bits, configurável);
  - Memória (RAM e ROM);
  - Banco de Registradores;
  - ULA:
    - Considere que a ULA executa qualquer operação lógica ou aritmética;
    - Quando acabar a analise de todas instruções, saberemos quais operações a ULA necessitará;
  - MUX.

# RTN

- Básico sobre RTN (register transfer notation):
  - Formaliza a descrição :
    - Do funcionamento. RTN abstrata: usa a arquitetura;
    - Da estrutura. RTN concreta: usa a organização:
      - Cada passo é um pulso de clock.
  - Misto de linguagem natural e expressões matemáticas:
    - Estabelece a relação entre o que está do lado esquerdo do símbolo com o que está do lado direito.
  - As transferências só acontecem no pulso de clock;
  - Considera que os sinais estão estáveis no momento do clock;
  - Considera que a lógica combinacional tem resposta instantânea.

# Símbolos e significados em RTN:

| Símbolo | Significado |
|---|---|
| <- | Transferência entre registradores: $Reg_{esquerda}$ recebe o valor do $Reg_{direita}$ . |
| [ ] | Índice de Word: seleciona uma palavra, ou a faixa de palavras, da memória indicada. |
| < > | Índice de Bit: seleciona a faixa de bits ou um bit da memória indicada. |
| n...m | Índice de Faixa: indica a faixa da esquerda para a direita (pode ser decrescente). |
| -> | If-then: verdadeiro na esquerda implica receber valor ou aplicar ação na direita. |
| := | Definição: ajuda a documentar. ld(opcode:=3) → R[ra] ← M[MA] |
| # | Concatenação de bits. |
| : | Separador de ações que acontecem em paralelo. |
| ; | Separador de ações que acontecem em sequência: da esquerda para direita. |
| @ | Repete o número de vezes, da direita, o que está na esquerda (concatena). |
| { } | Modificador de operação. |
| ( ) | Agrupamento de operações ou valores. |
| = ≠ < ≤ ≥ > | Operadores de comparação, retorna valores binários (≠ pode ser: !=). |
| + - * / | Operadores aritméticos. |
| ^ v ¬ ⊕ ≡ | Operadores lógicos: and, or, not, xor, equivalência. Pode usar: & \| ! |

- Exemplos:
  - Flip-flop: A ← B
  - Registrador: A<m..1> ← B<m..1>
  - Incremento do PC: PC ← PC + 4
  - Carrega instrução da memória:
    - RTN abstrata: IR ← M[PC]
    - RTN concreta: MAd ← PC : C ← PC + 4
      MD ← M[MAd] : PC ← C
      IR ← MD
  - Extensão de sinal de 16 para 32 bits, do Reg1:
    - (16@Reg1<15>#Reg1<15..0>)
  - Seleção de registrador (rn) no banco de regs.:
    - A ← (rn = 0 → 0 : rn != 0 → R[rn] )

# Análise

- Utilizando a definição da instrução “add”*:
  - Defina os blocos construtivos necessários para executá-la;
  - Descreva, em RTN, o fluxo entre esses blocos;
  - Para cada comando RTN:
    - Conecte os blocos construtivos necessários;
    - Para que a instrução execute;
    - E anote os pontos de controle utilizados.
  - Faça um rascunho do fluxo de dados;
  - Simule manualmente (papel e caneta).

*A definição das instruções pode ser encontrada:
- No greencard;
- Na página de atividades (links úteis).

- Quando o FD estiver pronto:
  - Verifique se atende o funcionamento de todas as outras instruções, do tipo R, que implementaremos:
    - Subtração (sub);
    - E lógico (and);
    - OU lógico (or);
    - Comparação Menor Que (set if less than: slt).
- Crie uma tabela com as instruções e os pontos de controle:
  - Descrevendo o estado de cada ponto de controle;
  - Para cada instrução implementada.

- Exemplo de tabela:
  - Com os sinais de controle;
  - E as instruções.

| Instrução | CtrlR1 | CtrlR2 | OpALU | RegWr | MemRd | MemWr |
|---|---|---|---|---|---|---|
| **add** | **1** | **0** | **001** | **1** | **0** | **0** |
| **sub** | **0** | **1** | **010** | **1** | **0** | **0** |
| **...** | **...** | **...** | **...** | **...** | **...** | **...** |
| **slt** | **0** | **1** | **100** | **1** | **0** | **0** |

- Agora que o FD atende as instruções R:
  - Implemente em VHDL;
  - Verifique o funcionamento usando simulação:
    - Use a tabela com os pontos de controle.
  - Implemente no kit de desenvolvimento:
    - E faça os testes necessários.

Insper

www.insper.edu.br