Aula 04 - 25/11/22

▼ Instruções de acesso á memória.

→ Armazenar / ler dados que não cabem nos registradores

- A memória é como um vetor: cada posição corresponde a 1 byte e possui um endereço.
- Todo dado armazenado na memória deve ser armazenado num endereço múltiplo de 4 (tamanho de uma palavra: 4 bytes = 32 bits) → Chamamos isso de restrição de alinhamento.
- As instruções em assembly mips são:

1. lw reg, offset (base) → lw = load word, esta instrução carrega um dado da memória do endereço **base +offset** no registrador **reg**.

   Obs: base é um registrador e offset, um número constante.

2. sw reg, offset (base) → sw = store word, esta instrução escreve o dado do registrador **reg** na memória, no endereço **base +offset**.
3. Syscall código 9: alocação dinâmica de memória

   → $a0: tamanho a ser alocado em bytes.

   → $v0: retorna o endereço base.

   ▼ Exemplos

   ▼ g = h + A[8];

   → g => $s1, h => $s2, end base de A => $s3

   Em MIPS:

```
lw $t0, 32($s3) ; carregou em t0 o A[8]
add $s1, $s2, $t0 ; fez a soma do t0 com s2 e armazenou em s1.
```

▼ A[12] = g + A[3];

```
lw $t0, 16($s3) ; $t0 = A[4]
add $t1, $s1, $t0 ; $t1 = g +A[4]
sw $t1, 48($s3) ; salva a soma em A[12]
; apenas variáveis vão para registradores
```

▼ Vetores em Assembly

- tem endereços de memória sequenciais.
- Estão sempre na memória, o que é diferente dos registradores.
- acesso eficiente para achar a posição.
  - vetor ⇒ base end * 4 (pegamos a base do vetor e multiplicamos por 4 quando usamos no MIPS), por isso nos exemplos de instruções de acesso á memória usamos o valor da base do vetor multiplicado por 4.

▼ Instruções imediatas ***** PROVA → EXPLIQUE INSTRUÇÕES IMEDIATAS**

- São instruções que operam sobre um registrador e uma constante. Geralmente são variações de outras instruções, terminadas por “i”.
- São instruções como i++ de outras linguagens.
- Existe somente a addi, não tem a subi.
- Exemplos:
  - addi $s3, $s3, 4 #s3 = s3+4
  - addi $s2, $s1, -1 #s2 = s1 +(-1)
- Princípio de design 3: Torne o caso comum mais rápido.
  - O uso de constante pequenas é muito comum.
  - Essas instruções evitam uma instrução lw.
  - **Obs**: Evitar usar a constante 0 em instruções imediatas.

Representação de inteiros

Todas as instruções que vimos até aqui operam com inteiros de 32 bits.

- Há dois tipos de inteiros: com sinal e sem sinal

Inteiro sem sinal

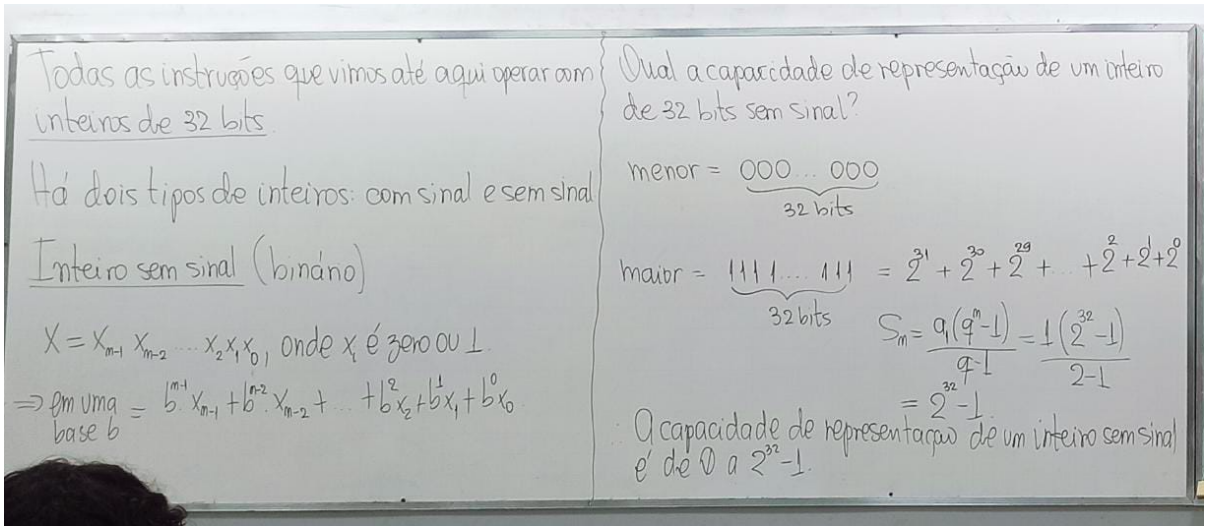
X = Xn-1Xn-2 … X2X1X0, onde Xi é 0 ou 1.

⇒ Em uma base b = $b^(n-1).Xn-1 + b^(n-2).Xn-2 + ... + b^2x2 + b^1x1 + b^0x0$.

Qual a capacidade de representação de um inteiro de 32 bits sem sinal?

menor = 000 … 000 } 32 bits.

maior = 111 … 111 } 32 bits = $2^(32) + 2^(31) + ... + 2^2 + 2^1 + 2^0$



A capacidade de representação de um inteiro sem sinal é de 0 a 2^(32) - 1.

Inteiro com sinal (binário)

Complemento a 2: dado x, -x = (não x) + 1.

⇒ bit mais significativo é o bit de sinal.

⇒ 0 corresponde a positivo enquanto o 1 a negativo.

⇒ um número de b bits x, |-x| = 2^b - x

Qual a capacidade de representação de um inteiro de 32 bits com sinal ??

1. MSB (Most Significant Bit) = 0 (positivo) ⇒ 2^31 combinações o que dá de 0 a 2^31 números.
2. MSB (Most Significant Bit) = 1 (negativo) ⇒ 2^31 combinações o que dá de -2^31 a -1

Portanto, a capacidade de representação de um inteiro com sinal de 4 bytes é -2^31 até 2^31 - 1.