

# Sistemas Hardware-Software

Aula 1 – Apresentação da Disciplina &  
Inteiros na CPU

**Engenharia**

Fabio Lubacheski

Maciel Calebe Vidal

Igor Montagne

Fábio Ayres

# **Critérios para Avaliação**

# Cálculo da média final (MF)

$$NS = 0,10*Atv + 0,20*AI + 0,30*AF + 0,40*Lab$$

$$NC = 0,10*Atv + 0,20*AI + 0,25*AF + 0,40*Lab + 0,05*MC$$

## CONDIÇÕES:

$$((AI + AF) / 2) \geq 4,0 \text{ E}$$

$$AI \text{ e } AF \geq 3,5 \text{ E}$$

$$Lab \geq 5,0$$

Atv: Atividades  
AI: Aval. Intermediária  
AF: Aval. Final  
Lab: laboratórios  
MC: Prova mutirão C

Se atendida as CONDIÇÕES:

$$MF = \max(NS, NC)$$

Se NÃO atendida as CONDIÇÕES:

$$MF = \min(Atv, AI, AF, Lab, C)$$

# Colaboração e Integridade Acadêmica

- Nas entregas espera-se que todos os envios sejam **seus e somente seus**;
- Você é incentivado a discutir suas tarefas com outros alunos (ideias), mas espera-se que **o que você entregar seja seu**;
- **NÃO é aceitável** copiar soluções de outros alunos ou copiar soluções da Web (incluindo ferramentas de IA);
- **Nosso objetivo é que \*VOCÊ\* aprenda o conteúdo para estar preparado para exames, entrevistas e para o futuro**

# Colaboração e Integridade Acadêmica

- Assim, por conta de tudo que foi explicado, **não repasse e nem copie um atividade/laboratório de outro aluno (mesmo de outro semestre);**
- É muito importante que o aluno saiba que **ele é o responsável pelo seu trabalho;**
- Se forem detectadas cópias de trabalhos, ou "cópia disfarçada", ambos são tratados como plágio, e **a autoria real é irrelevante, tanto o original como a cópia, receberão nota zero.**

# Exercícios práticos (atividades e labs)

- Série de exercícios práticos de implementação
- Complexidade crescente
- Testes automatizados **quando possível**
  - Facilitar correção
  - Criar espaços para conversar da matéria e esclarecer dúvidas
- Na computação, **erros não são falhas** — eles são simplesmente experiência....

# Ferramentas

- **GCC 9.3** (ou superior) -- C99
- Linux (Preferencialmente **Ubuntu 22.04**)
- **PC x86-64**

**Não há suporte a outros sistemas. Instalem direto ou usem uma VM. Se usar VM, veja se funciona com Smowl.**



# Aula!





# **Conversão binário $\Leftrightarrow$ decimal para inteiros sem sinal (unsigned)**

# Revisão: conversão binário -> decimal

Número **1010011**<sub>2</sub> (base 2) com 6 bits:

$$1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 83 \text{ (base 10)}$$

1. Cada dígito multiplica uma potência de 2
2. O dígito mais significativo é 1 (multiplica a maior potência)
3. O dígito menos significativo é 0 (multiplica a menor potência)

# Revisão: conversão Decimal -> Binário

Fazemos agora o caminho inverso: dividimos sucessivamente por 2 e guardamos o resto

**$83_{10}$**  (base 10)

# Revisão: hexadecimal

Os dois números abaixo são o mesmo? Se não qual o bit diferente?

1001110011101110

1001110111101110

# Revisão: hexadecimal

Os dois números abaixo são o mesmo?

$0x9CEE_{16}$

$0x9DEE_{16}$

# Revisão: hexadecimal

Os dois números abaixo são o mesmo?

$$0x9CEE_{16} = 1001110011101110_2$$

$$0x9DEE_{16} = 1001110111101110_2$$

**Objetivo:** facilitar a leitura de números binários

- agrupar 4 em 4 bits em um dígito que vai de 0 a 15
- letras para os dígitos maiores que 10



# **Representação de números inteiros na linguagem C**

# Tipos inteiros na linguagem C

- Todo tipo tem tamanho **fixo**.
- Um inteiro pode ter os seguintes tamanhos:


Tamanho em bytes	Tipo em C	Capacidade
1	char	256
2	short	65536
4	int	$2^{32}$
8	long	$2^{64}$



# Tipos inteiros sem sinal (**unsigned**)

Representação para números positivos somente (modificador **unsigned**). O valor do maior número é dado por  $2^w - 1$ , onde  $w$  é o número de bits do tipo.

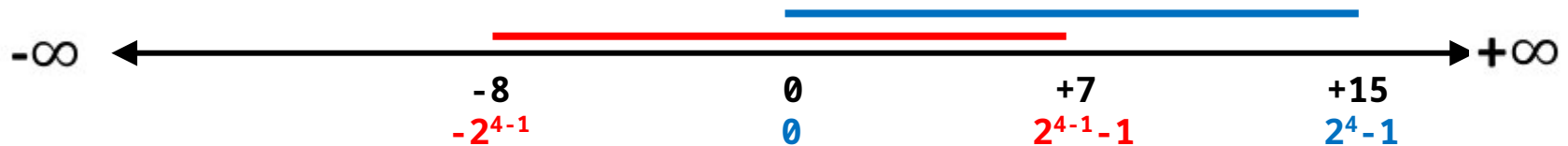
Tamanho em bytes	Tipo em C	Menor número	Maior Número
1 (8 bits)	<b>unsigned char</b>	0	255 ( $2^8 - 1$ )
2 (16 bits)	<b>unsigned short</b>	0	65535 ( $2^{16} - 1$ )
4 (32 bits)	<b>unsigned int</b>	0	$2^{32} - 1$
8 (64 bits)	<b>unsigned long</b>	0	$2^{64} - 1$



# **Como representar inteiros NEGATIVOS na linguagem C ?**

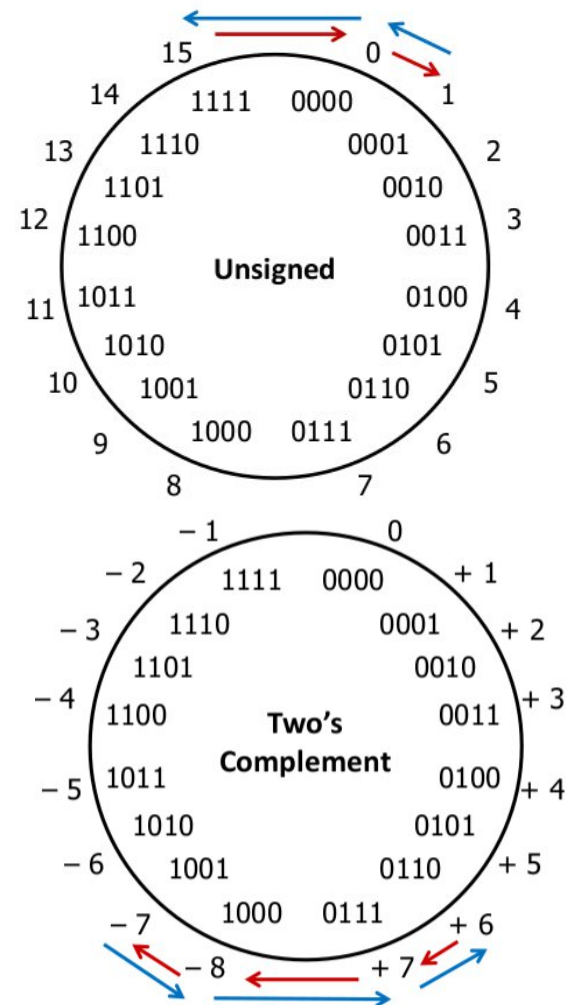
# Inteiros **unsigned** e **signed**

- O hardware (e a linguagem C) suporta dois tipos de números inteiros:
  - **unsigned** – não negativos  $0 \dots 2^w - 1$
  - **signed** – negativos e positivos  $-2^{w-1} \dots 2^{w-1} - 1$
- Exemplo: Considere um inteiro de **4-bits** teríamos os seguintes valores.

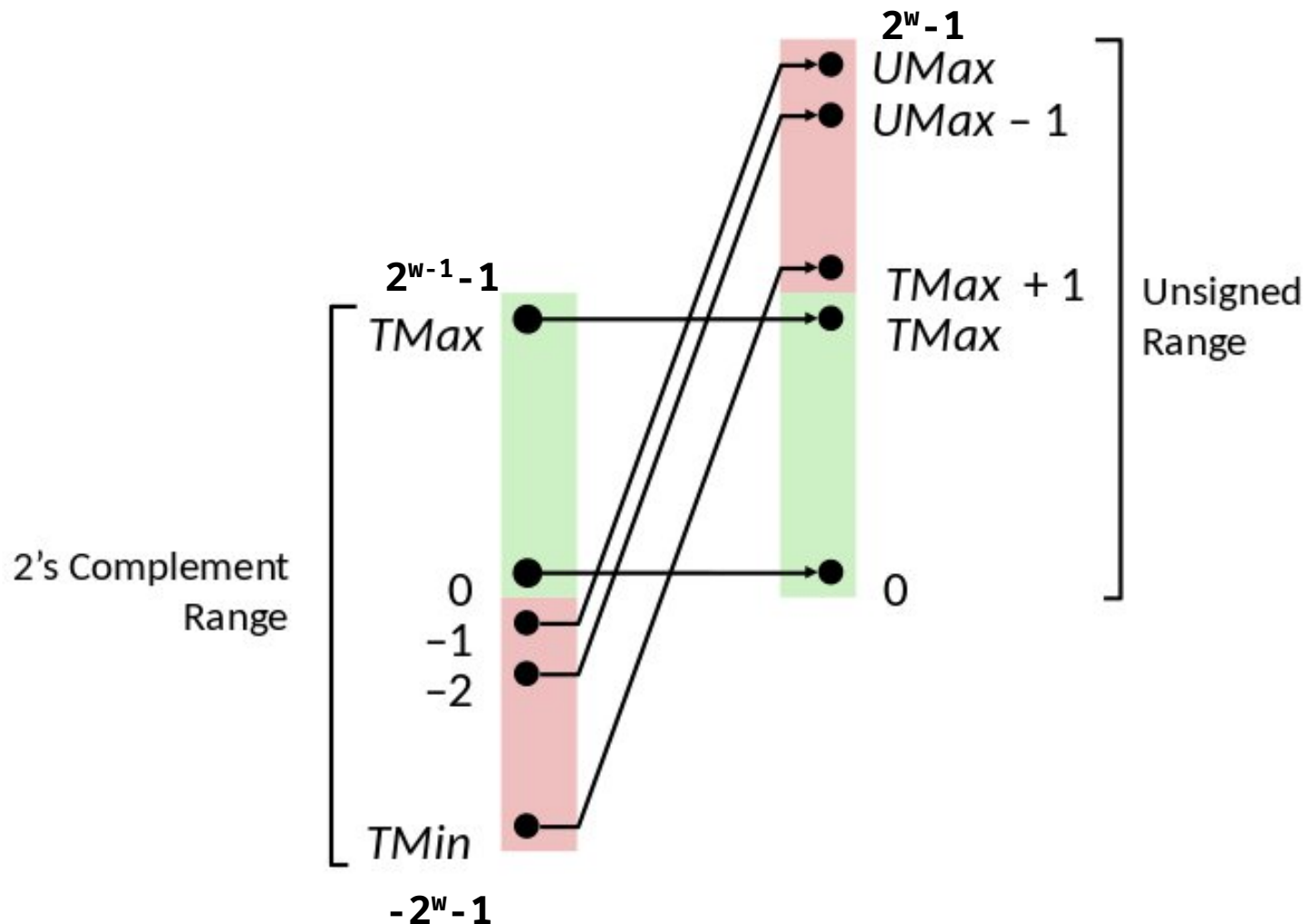


# Inteiros unsigned e signed

Bits	Unsigned	Signed
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	-8
1001	9	-7
1010	10	-6
1011	11	-5
1100	12	-4
1101	13	-3
1110	14	-2
1111	15	-1



# Conversões entre inteiros signed/unsigned





# **Conversão binário $\Leftrightarrow$ decimal para inteiros com sinal (signed)**

# Inteiros com sinal (Complemento de dois)

Dado um inteiro  $\mathbf{b_2}$  com  $\mathbf{w}$  **bits**, seu valor em decimal é

$$b_{10} = -2^{w-1} b_{w-1} + \sum_{i=0}^{w-2} 2^i b_i$$

1. Somamos todos os bits normalmente
2. Menos o último, que ao invés de somar **subtrai**

# Binário sem sinal - Exercício

Considere o trecho de código abaixo:

```
char unsigned x = 0b10110001;
```

Qual o valor de ch ?



# Binário sem sinal - Exercício

Considere o trecho de código abaixo:

```
char unsigned x = 0b10110001;
```

Qual o valor de ch ?

$$2^7 + 2^5 + 2^4 + 2^0 =$$

$$128 + 32 + 16 + 1 = 177 \text{ (base 10)}$$

# Binário com sinal - Exercício

Considere o trecho de código abaixo:

```
char x = 0b10110001;
```

Qual o valor de ch ?

# Binário com sinal - Exercício

Considere o trecho de código abaixo:

```
char x = 0b10110001;
```

Qual o valor de ch ?

$$-2^7 + 2^5 + 2^4 + 2^0 =$$

$$-128 + 32 + 16 + 1 = -79 \text{ (base 10)}$$

# Decimal com sinal - Exercício

Considere o trecho de código abaixo:

```
char x = -14;
```

Qual o valor de ch em binário (base 2) ?

Para obter a representação negativa de qualquer número inteiro faz-se o complemento bit a bit e em seguida, adicionando um!

$$( \sim x + 1 == -x )$$

# Decimal com sinal - Exercício

Considere o trecho de código abaixo:

```
char x = -14;
```

Qual o valor de ch em binário (base 2) ?

$$( \sim x + 1 == -x )$$

$$14_{10} = 00001110_2$$

$$\sim 00001110_2 = 11110001_2$$

$$11110001_2 + 1 = 11110010_2$$

# Conversões entre tipos inteiros

Duas regras:

1. O valor é mantido quando convertemos de um tipo menor para um tipo maior

**char -> int**

2. A conversão de um tipo maior para um tipo menor é feita pegando o X bits menos significativos.

**int -> char** pega os 8 bits menos significativos, o restante é descartado

# Deslocamento de bits (shift)

- Shift para esquerda ( $x \ll n$ ) é equivalente a multiplicar  $x$  por  $2^n$
- Shift para direita ( $x \gg n$ ) é equivalente a dividir  $x$  por  $2^n$
- A operação de shift é **mais rápida** do que a multiplicação e divisão !

	x	0010	0010
	$x \ll 3$	0001	0000
logical:	$x \gg 2$	0000	1000
arithmetic:	$x \gg 2$	0000	1000

	x	1010	0010
	$x \ll 3$	0001	0000
logical:	$x \gg 2$	0010	1000
arithmetic:	$x \gg 2$	1110	1000

# Atividade prática

## Conversão de números: bases e sinal

1. rodar programa bases\_e\_sinais
2. colocar sua solução em solucao.txt
3. verificar se tudo está ok rodando

```
./bases_e_sinais < solucao.txt
```



# Atividade Extra (Não será cobrada)

Atividade extra para os curiosos!

Pesquise como o computador representa números reais.  
Qual o padrão utilizado?

# Git

<https://insper.github.io/SistemasHardwareSoftware/>

<https://github.com/Insper/SistemasHardwareSoftware>

# Insper

[www.insper.edu.br](http://www.insper.edu.br)