Insper

Sistemas Hardware-Software

Aula 1 – Apresentação da Disciplina & Inteiros na CPU

Ciência da Computação

Carlos Menezes Maciel Calebe Vidal Igor Montagne Fábio Ayres

Professor

Carlos Menezes

Objetivo da disciplina

Resumidamente o objetivo da disciplina é apresentar as interfaces e abstrações que permitem entender:

- Como seu código-fonte se torna algo que o computador entenda ?
- O que acontece quando seu computador executa um ou mais processos?

Código-fonte em muitas formas

```
if (x != 0) y = (y+z)/x;

Compiler
```

```
cmpl $0, -4(%ebp)
je .L2
movl -12(%ebp), %eax
movl -8(%ebp), %edx
leal (%edx,%eax), %eax
movl %eax, %edx
sarl $31, %edx
idivl -4(%ebp)
movl %eax, -8(%ebp)
.L2:
```

Linguagem de alto nível (linguagem C) – mais amigável para nós!

Linguagem Assembly

Assembler

Código de máquina - Hardware executa strings de bits

Insper

Conteúdo da disciplina

Linguagem C car *c = malloc(sizeof(car)); c->miles = 100; c->gals = 17; float mpg = get_mpg(c); free(c);

```
Linguagem
Assembly

get_mpg:

pushq %rbp
movq %rsp, %rbp
...
popq %rbp
ret
```

Código de Máquina

1100000111111101000011111

Decimal & Binário & Hexa Assembly do x86-64

Executáveis

funções & stacks

Arrays & structs

Padrão POSIX

Alocação de memória

Sinais & função Exec

Arquivos

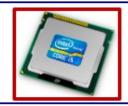
Processos & Threads

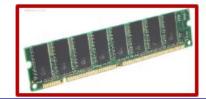
Sincronização

Sistema Operacional



Computador







Ao final da disciplina você poderá

- Entender como realmente funciona a execução de um programa;
- Compreender as abstrações que existem no hardware em que o programa é executado;
- Depurar melhor o código-fonte implementado;
- Avaliar e melhorar o desempenho na execução dos programas.
- **❖** E por fim, como construir programas mais eficientes.

Aulas

- · Quartas-feiras, 15h45 às 17h45
- Sextas-feiras, 15h45 às 17h45

Atendimento presencial

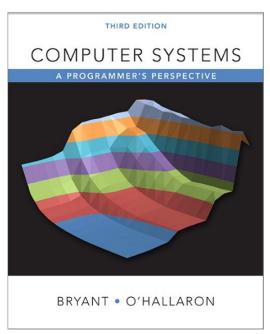
• Sextas-feiras, 14h15 às 15h45

Bibliografia básica

Computer Systems: A Programmer's Perspective

- R. E. Bryant and D. R. O'Hallaro
- site: http://csapp.cs.cmu.edu

Este livro é **realmente importante** para disciplina!



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 C
```

CONDIÇÕES: ((AI + AF) / 2) >= 4,0 E PI e PF >= 3,5 E

L >= 5.0

Se atendidas as CONDIÇÕES: MF = max(NS, NC)

Se NÃO atendidas as CONDIÇÕES: MF = min(Atv, AI, AF, Lab, C)

Se MF >= 6,0 então APROVADO

Atv: Atividades

Al: Aval. Intermediária

AF: Aval. Final

Lab: Laboratórios

C: Prova mutirão 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

Exercícios práticos (entrega)

- Github classroom
 - Testes automatizados para alguns exercícios
 - Ver link e tutorial em Conteúdos (Blackboard) para cadastro

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 proctorio.

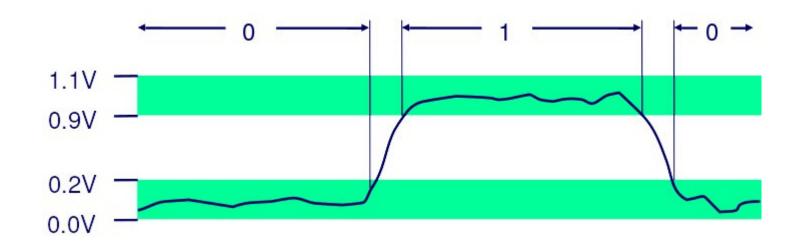
Aula!

Qual a saída do programa testa_bit.c?

```
int main(){
   char ch = 0b10000111;
   printf("ch:[%d]\n",ch);
   return 0;
}
```

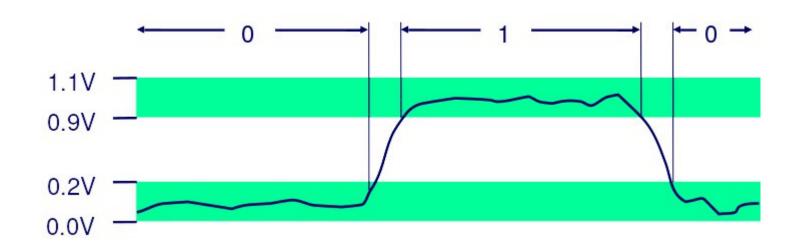
Representação de inteiros na CPU

Informação é codificada como sequência de 0 e 1



- Inteiros, Strings, Números reais
- Instruções da CPU, Endereços, etc

Informação é codificada como sequência de 0 e 1

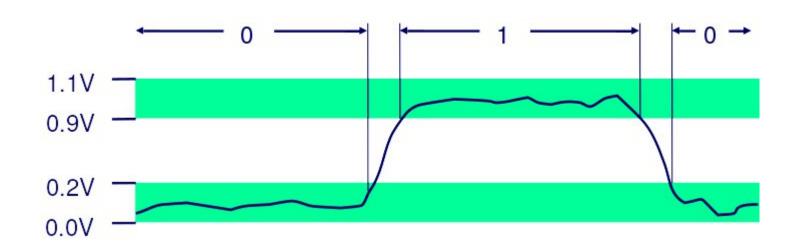


- Inteiros, Strings, Números reais
- Instruções da CPU, Endereços, etc

Não é possível distinguir conteúdo a partir de uma sequência de bits

Agrupamos 8 bits em 1 byte

Informação é codificada como sequência de 0 e 1

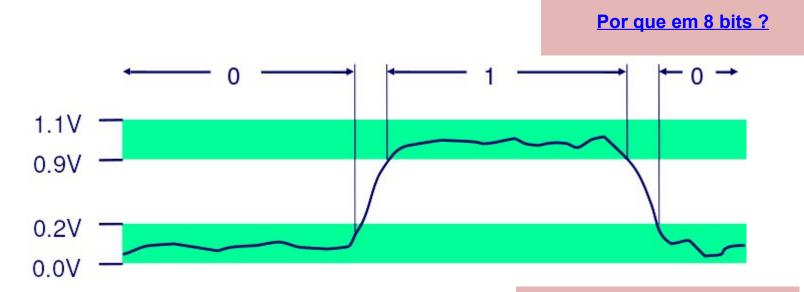


- Inteiros, Strings, Números reais
- Instruções da CPU, Endereços, etc

Não é possível distinguir conteúdo a partir de uma sequência de bits

Agrupamos 8 bits em 1 byte

Informação é codificada como sequência de 0 e 1



- Inteiros, Strings, Números reais
- Instruções da CPU, Endereços, etc

Não é possível distinguir conteúdo a partir de uma sequência de bits

Conversão binário <=> decimal para inteiros sem sinal (unsigned)

Inteiros (decimal)

Número 9153



Inteiros (decimal)

Número **9153**

$$9000+100+50+3 = 9\times10^3+1\times10^2+5\times10^1+3\times10^0$$

- 1. Cada dígito multiplica uma potência de 10
- 2. O dígito mais significativo é 9 (multiplica a maior potência)
- 3. O dígito menos significativo é 3 (multiplica a menor potência)

Inteiros (binário)

Número **1010011**₂(base 2) com 6 bits:

Inteiros (binário)

Número **1010011**₂ (base 2) com 6 bits:

$$1x2^6+0x2^5+1x2^4+0x2^3+0x2^2+1x2^1+1x2^0 = 83$$
 (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)

Conversão Binário -> Decimal: Exercício

Converta o número abaixo para decimal

110000102

Conversão Decimal -> Binário

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

75₁₀ (base 10)

Conversão Decimal -> Binário: Exercício

Agora é sua vez:

165₁₀

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

1001110011101110

1001110111101110

Os dois números abaixo são o mesmo?

0x9CEE₁₆

0x9DEE₁₆

Os dois números abaixo são o mesmo?

```
0 \times 9 \subset EE_{16} = 1001110011101110_2
```

$$0 \times 9DEE_{16} = 1001110111101110_2$$

Objetivo: facilitar a leitura de números binários

Os dois números abaixo são o mesmo?

```
0 \times 9CEE_{16} = 1001110011101110_2

0 \times 9DEE_{16} = 1001110111101110_2
```

Ideia:

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

Binário	Hexa	Binário	Hexa
0000	0×0	1000	0x8
0001	0×1	1001	0×9
0010	0×2	1010	0xA
0011	0×3	1011	0xB
0100	0×4	1100	0xC
0101	0×5	1101	0xD
0110	0×6	1110	0×E
0111	0×7	1111	0xF

Exercício

Converta para binário: 0xDE9 (base 16)

Converta para hexadecimal: 1100 1110 0011 1010 (base 2)

Exercício

Converta para binário: 0xDE9 (base 16)

1101 1110 1001 (base 2)

Converta para hexadecimal: 1100 1110 0011 1010 (base 2)

0xCE3A (base 16)

Codificação de números inteiros

Tipos inteiros na linguagem C

- Todo dado 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

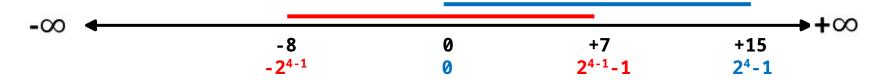
Tipos inteiros sem sinal (unsigned)

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)	unsigned char	0	255 (2 ⁸ -1)
2 (16 bits)	unsigned short	0	65535 (2 ¹⁶⁻ 1)
4 (32 bits)	unsigned int	0	2 ³² - 1
8 (64 bits)	unsigned long	0	2 ⁶⁴ - 1

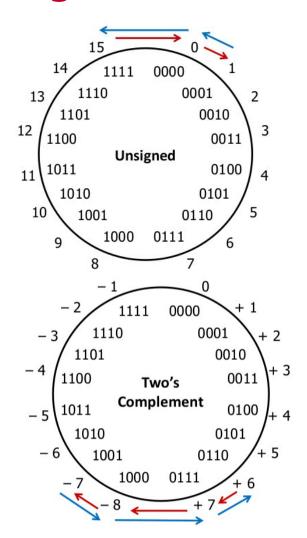
Inteiros unsigned e signed

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

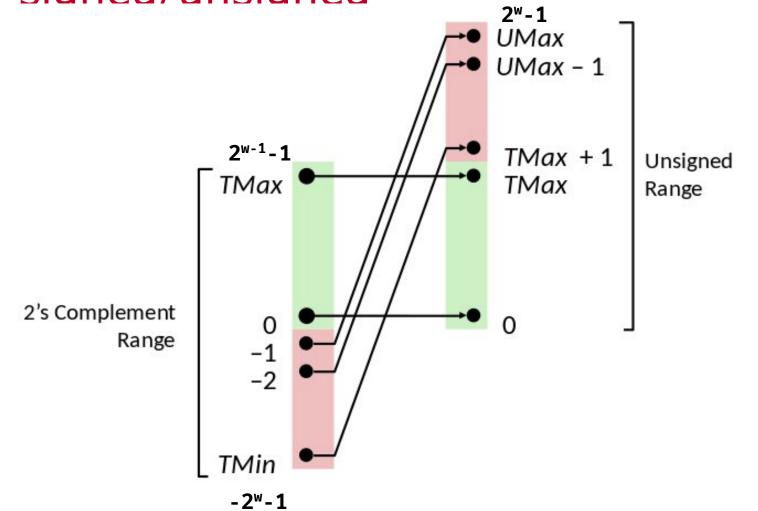


Inteiros unsigned e signed

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 sianed/unsianed



Conversão binário <=> decimal para inteiros com sinal (signed)

Inteiros com sinal (Complemento de dois)

Dado um inteiro b₂ com 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 ser somado, é subtraído

Considere o trecho de código abaixo:

char unsigned x = 0b10110001;

Considere o trecho de código abaixo:

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

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

128 + 32 + 16 + 1 = 177 (base 10)

Considere o trecho de código abaixo:

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

Considere o trecho de código abaixo:

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

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

-128 + 32 + 16 + 1 = -79 (base 10)

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!

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

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 <u>esquerda</u> (x<<n) é equivalente a <u>multiplicar</u>
 x por 2ⁿ
- Shift para <u>direita</u> (x>>n) é equivalente a <u>dividir</u> x por 2ⁿ
- A operação de shift é mais rápida do que a multiplicação e divisão !

	Х	0010	0010
	x<<3	0001	0000
logical:	x>>2	0000	1000
arithmetic:	x>>2	00 00	1000

	Х	1010	0010
	x<<3	0001	0000
logical:	x>>2	0010	1000
arithmetic:	x>>2	11 10	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</pre>

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/SistemasHardwareSoftwareBCC/

https://github.com/Insper/SistemasHardwareSoftwareBCC/



Insper

www.insper.edu.br