Insper

Sistemas Hardware-Software

Aula 1 – Apresentação da Disciplina &
Inteiros na CPU

Engenharia
Fabio Lubacheski
Maciel Calebe Vidal
Igor Montagne
Fábio Ayres

Professores

- Fabio Lubacheski

- Diego Saragoza da Silva

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:
```

Assembler

```
10000011011111000010010000011100000000000
0111010000011000
10001011010001000010010000010100
10001011010001100010010100010100
10001101000001000000010
1000100111000010
11000001111110100011111
11110111011111000010010000011100
10001001010001000010010000011000
```

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

Linguagem Assembly

Código de máquina - Hardware executa strings de bits

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

```
0111010000011000
10001101000010000000010
1000100111000010
11000001111110100011111
```

Computador



Sistema Operacional

- Decimal & Binário & Hexa
- Assembly do x86-64
- Executáveis
- funções & stacks
- Arrays & structs
- Padrão POSIX
- Alocação de memória
- Sinais & função Exec
- Arquivos
- Processos & Threads
- Sincronização

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

- Seg 15h45 às 17h45 – Lab Ágil 2
- Qui 15h45 às 17h45 – Lab Ágil 2

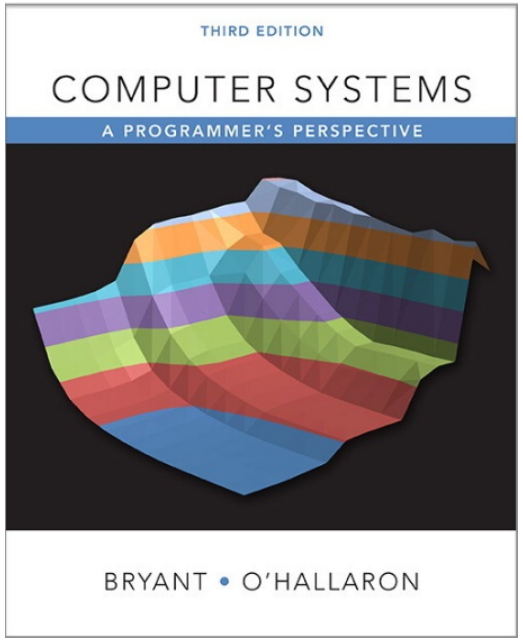
Atendimento presencial

- Qui 14h00 às 15h30 – Lab Ágil 2

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) >= 4,5 E
AI e AF >= 3,5 E
Lab >= 5,0

Se atendida as CONDICOES:
MF = max(NS, NC)

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

Se MF >= 6,0 então APROVADO

Atv: Atividades
AI: 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!

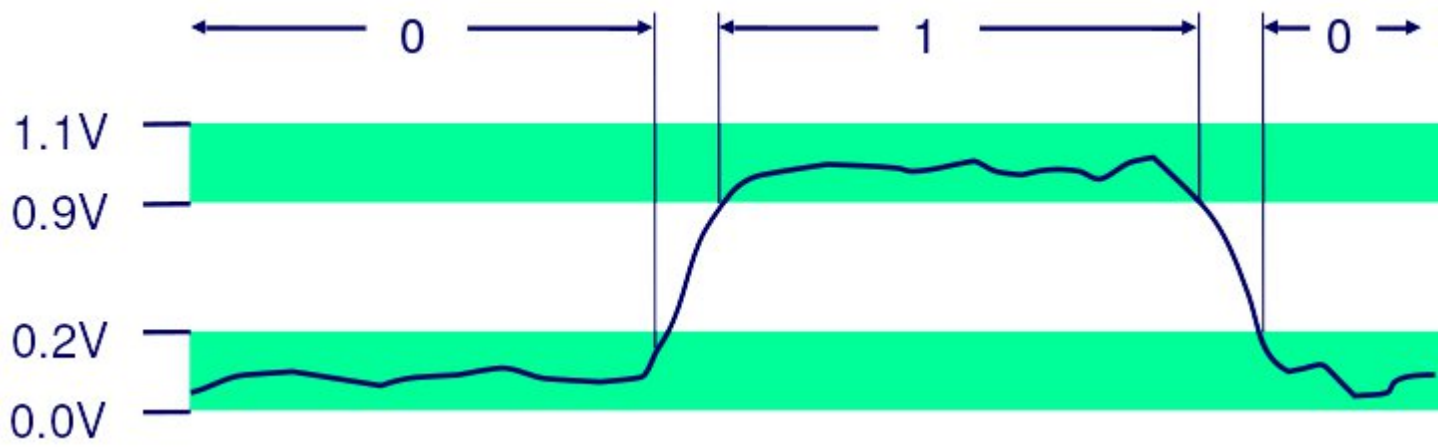
Qual a saída do programa **testa_bit.c** ?

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

Representação de inteiros na CPU

Bits e Bytes

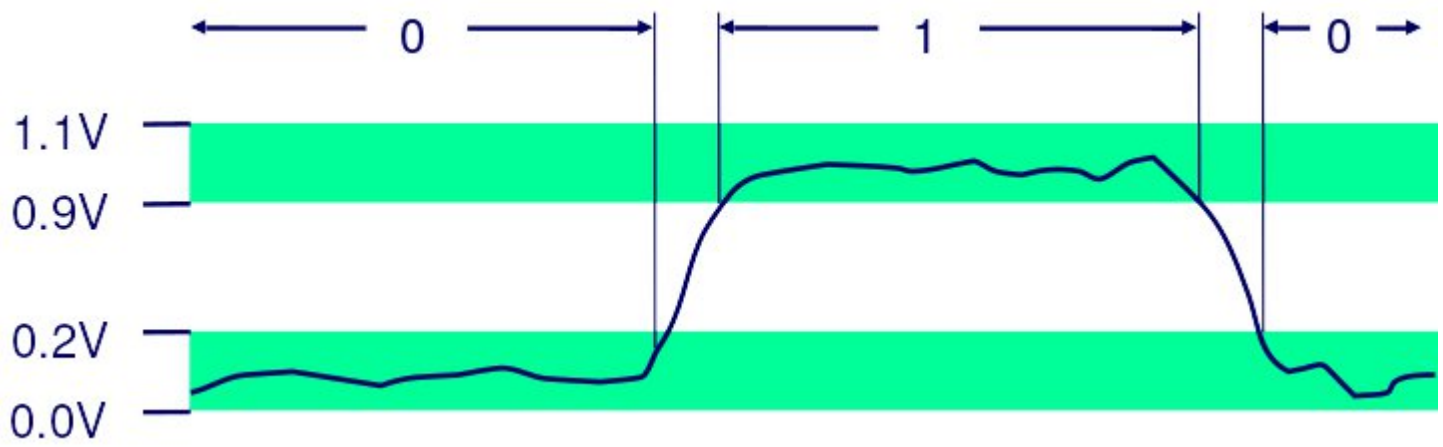
Informação é codificada como sequência de 0 e 1



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

Bits e Bytes

Informação é codificada como sequência de 0 e 1



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

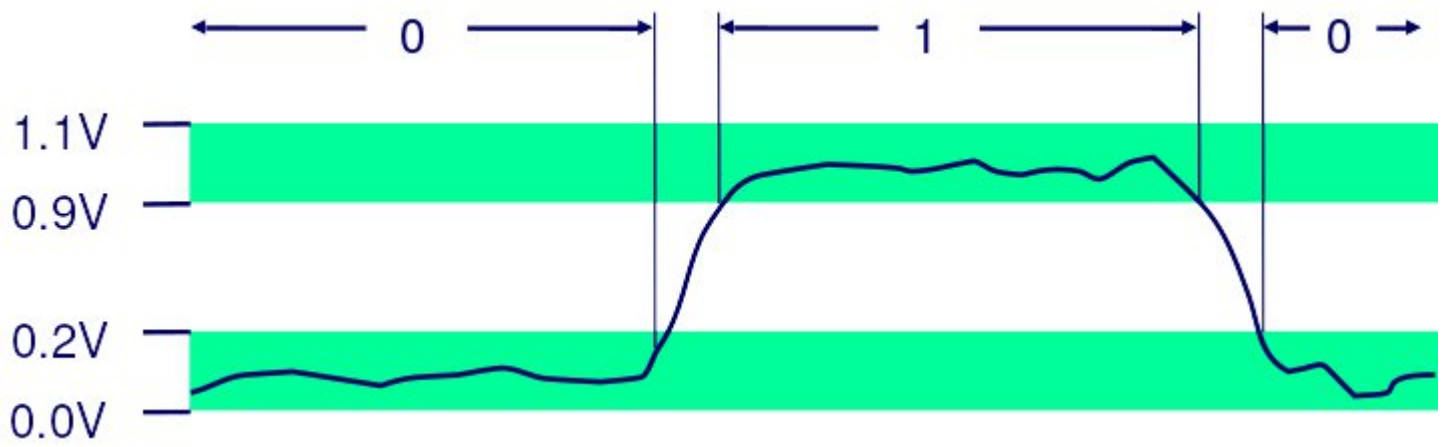
Não é possível distinguir conteúdo a partir de uma sequência de bits

Insper

Bits e Bytes

Agrupamos 8 bits em 1 byte

Informação é codificada como sequência de 0 e 1



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

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

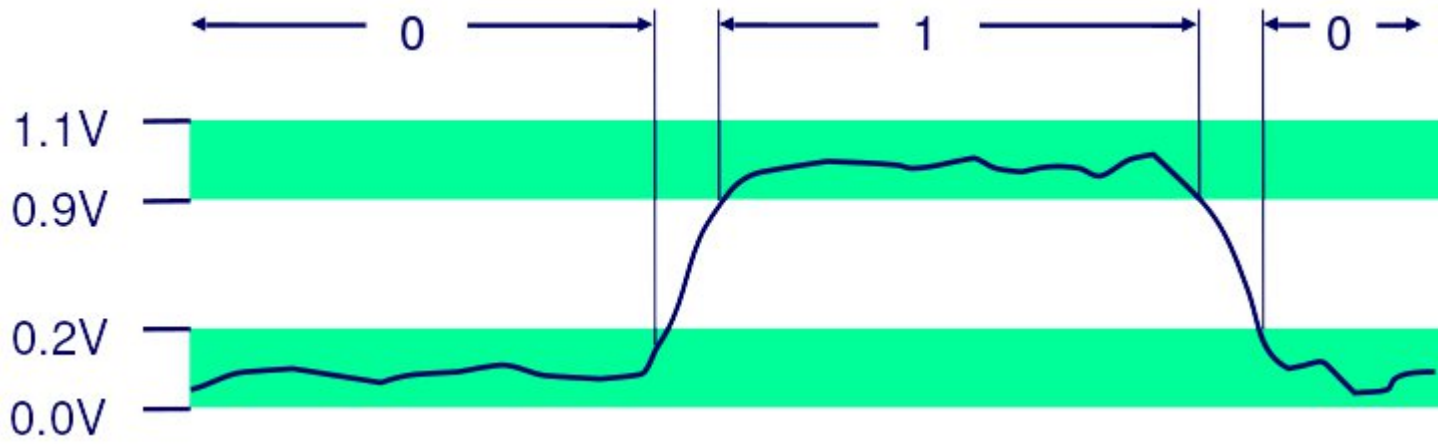
Insper

Bits e Bytes

Agrupamos 8 bits em 1 byte

Informação é codificada como sequência de 0 e 1

Por que em 8 bits ?



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

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

Insper

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

Inteiros (decimal)

Número **9153**

Inteiros (decimal)

Número **9153**

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

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 $\mathbf{1010011_2}$(base 2) com 6 bits:

Inteiros (binário)

Número $\mathbf{1010011_2}$ (base 2) com 6 bits:

$$\mathbf{1}x2^6+\mathbf{0}x2^5+\mathbf{1}x2^4+\mathbf{0}x2^3+\mathbf{0}x2^2+\mathbf{1}x2^1+\mathbf{1}x2^0 = \mathbf{83}\ (\text{base } 10)$$

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)

Conversão Binário -> Decimal: Exercício

Converta o número abaixo para decimal

$$11000010_2$$

Conversão Decimal -> Binário

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

$$\mathbf{75_{10}}\ (\text{base } 10)$$

Conversão Decimal -> Binário: Exercício

Agora é sua vez:

$$165_{10}$$

Hexadecimal

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

```
1001110011101110

1001110111101110
```

Hexadecimal

Os dois números abaixo são o mesmo?

$0x9CEE_{16}$

$0x9DEE_{16}$

Hexadecimal

Os dois números abaixo são o mesmo?

$0x9CEE_{16} = 1001110011101110_2$

$0x9DEE_{16} = 1001110111101110_2$

Objetivo: facilitar a leitura de números binários

Hexadecimal

Os dois números abaixo são o mesmo?

$0x9CEE_{16} = 1001110011101110_2$

$0x9DEE_{16} = 1001110111101110_2$

Ideia:

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

Hexadecimal

Binário	Hexa	Binário	Hexa
0000	0x0	1000	0x8
0001	0x1	1001	0x9
0010	0x2	1010	0xA
0011	0x3	1011	0xB
0100	0x4	1100	0xC
0101	0x5	1101	0xD
0110	0x6	1110	0xE
0111	0x7	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}

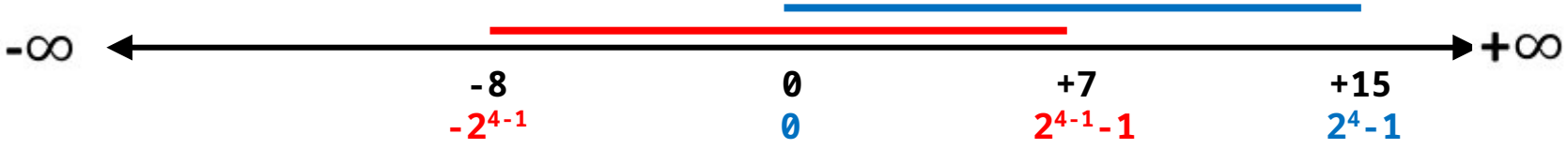
Tipos inteiros sem sinal (**unsigned**)

Representação para números positivos somente (modificador **unsigned**). O valor do maior número é dado por $\mathbf{2^w-1}$, 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**

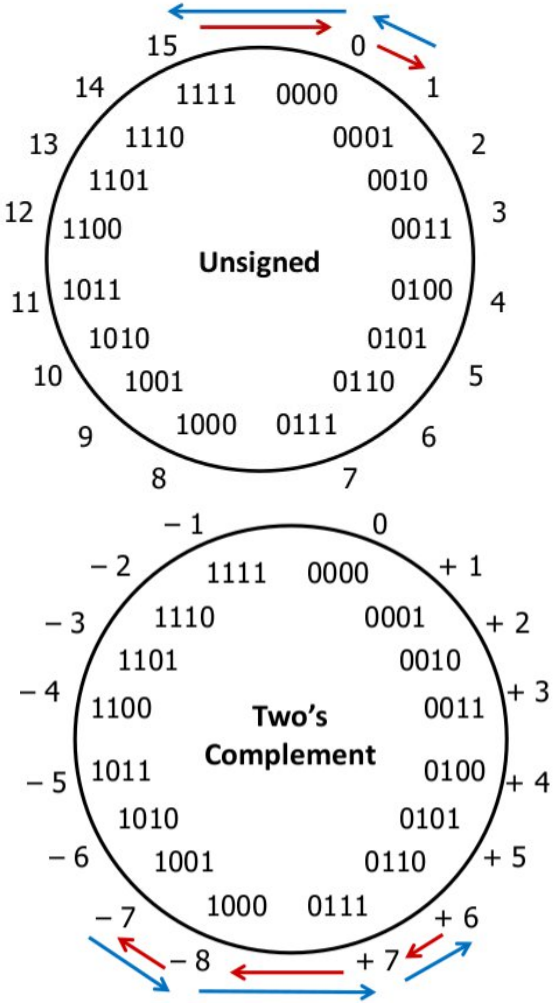
- O hardware (e a linguagem C) suporta dois tipos de números inteiros:
  - **unsigned** – não negativos **0 ... 2^w–1**
  - **signed** – negativos e positivos **-2^{w-1}… 2^{w-1}–1**
- Exemplo: Considere um inteiro de **4-bits** teríamos os seguintes valores.



Insper

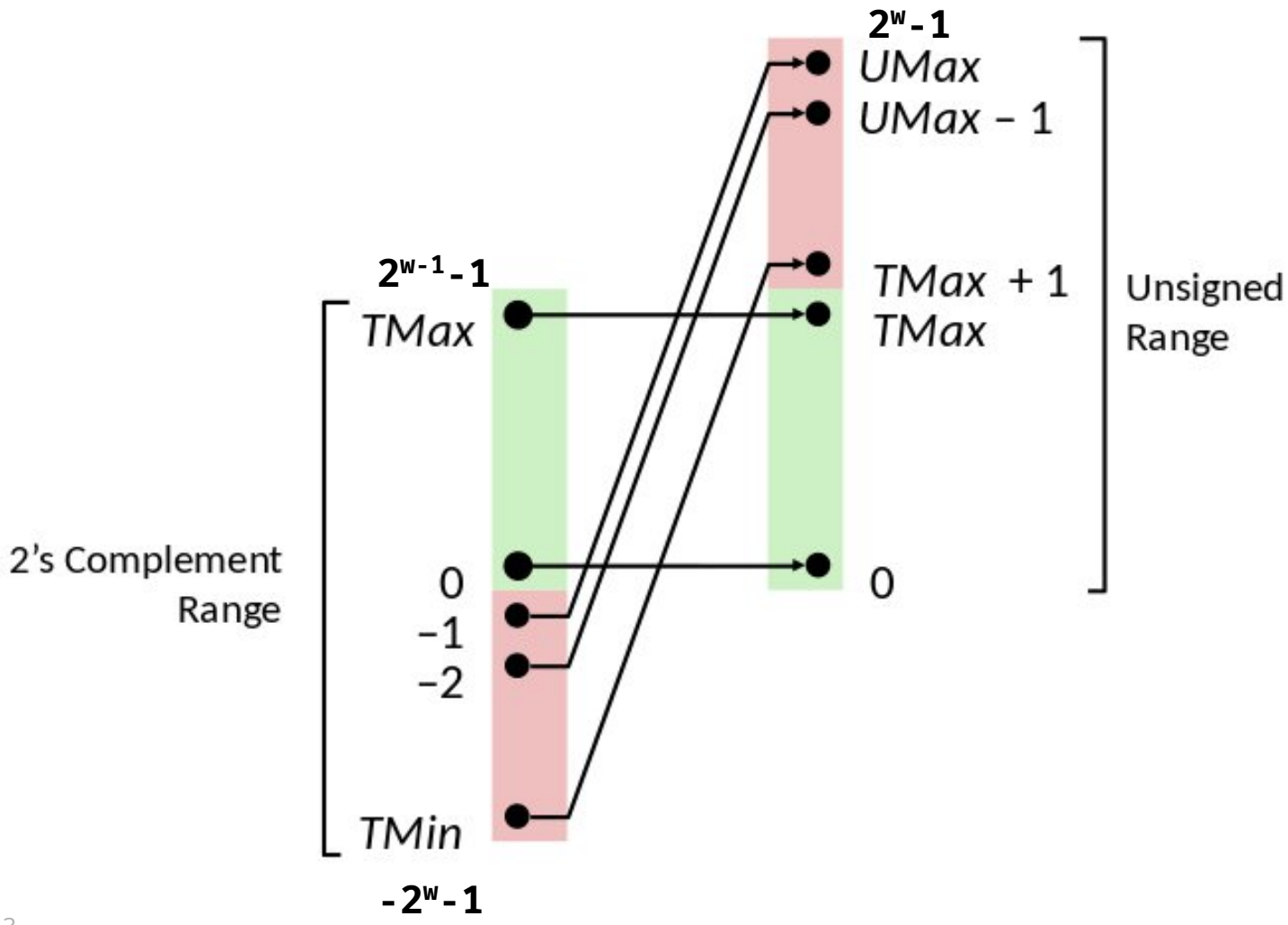
Inteiros unsigned e signed

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



Insper

Conversões entre inteiros signed/unsigned



Insper

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

Inteiros com sinal (Complemento de dois)

Dado um inteiro **b_2** 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 somar **subtrai**

Inteiros sem sinal - Exercício

Considere o trecho de código abaixo:

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

Qual o valor de ch ?

Inteiros 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 (base 10)

Inteiros com sinal - Exercício

Considere o trecho de código abaixo:

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

Qual o valor de ch ?

Inteiros 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 (base 10)

Inteiros 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!

(~x + 1 == -x)

Inteiros com sinal - Exercício

Considere o trecho de código abaixo:

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

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

(~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<<n**) é equivalente a **multiplicar x** por 2^n
- Shift para **direita** (**x>>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<<3	0001 0000
logical:	x>>2	0000 1000
arithmetic:	x>>2	0000 1000

	x	1010 0010
	x<<3	0001 0000
logical:	x>>2	0010 1000
arithmetic:	x>>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