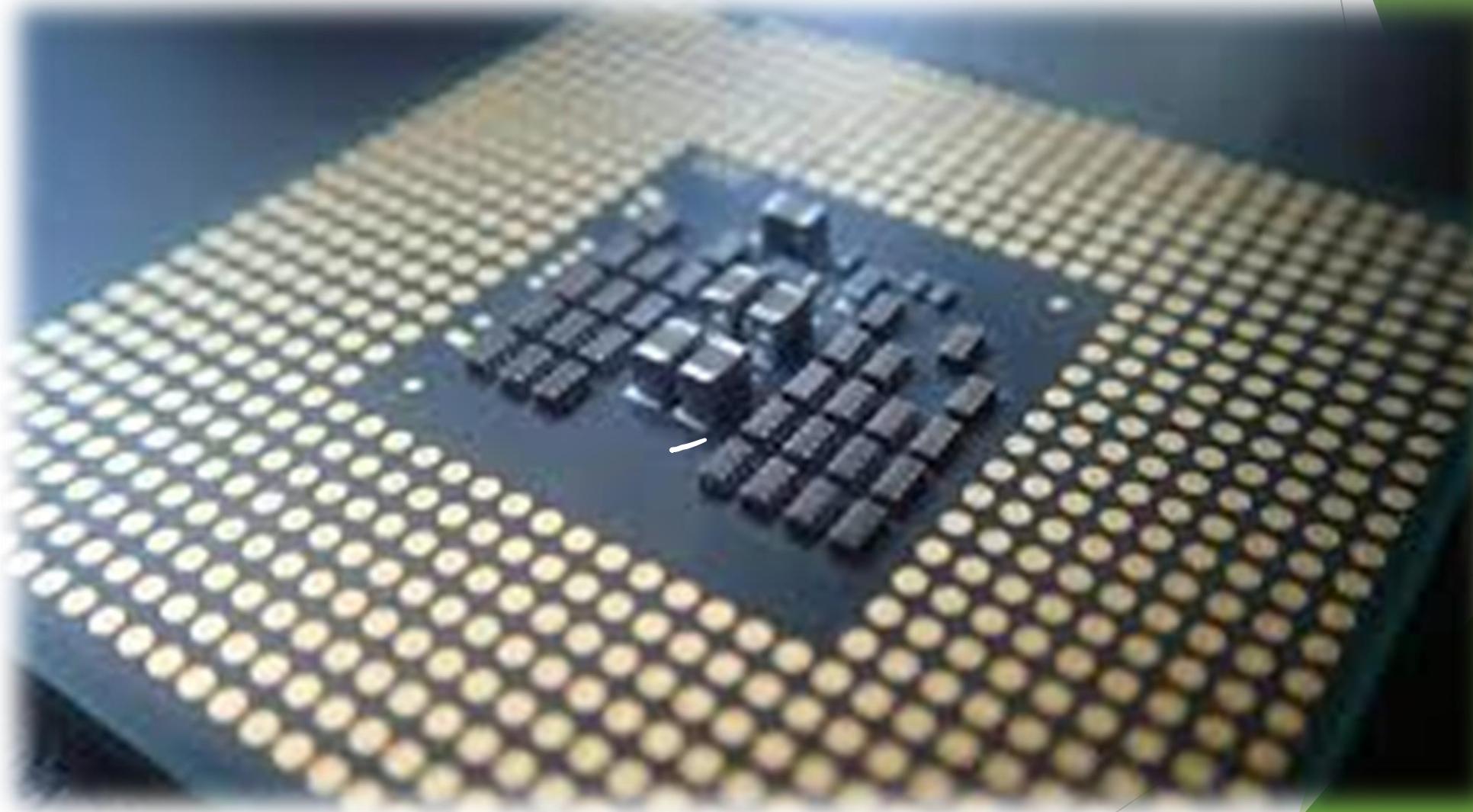


1 km =

~~1 km~~
1 metro

1.000 . 000 . 000

Processador



Processador

Qual a função de um processador?

Processador

O processador, também chamado de CPU (*Central Processing Unit*) ou UCP (Unidade Central de Processamento), é o componente de hardware responsável por processar dados e transformar em informações.

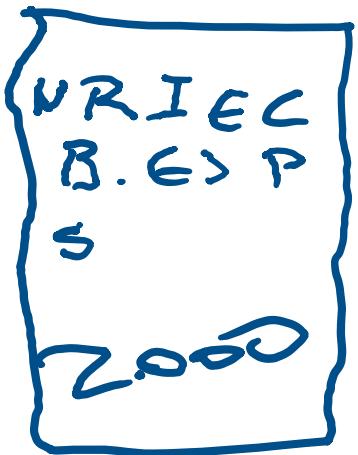
Ele também transmite estas informações para a placa mãe, que por sua vez as transmite para outros componentes (como o monitor, impressora, outros dispositivos).

Processador

Dados x Informação

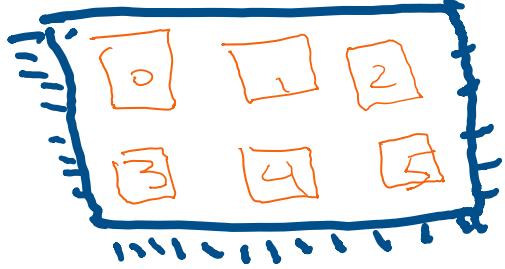
Processador

Dados x Informação



Processador

A placa mãe serve de ponte entre o processador e os periféricos. Outras funções do processador são fazer cálculos e tomar decisões lógicas, ou seja é o ‘cérebro’ do computador.



Processador

Características do processador:

Frequência de Processador (Velocidade, clock). Medido em hertz, define a capacidade do processador em processar informações ao mesmo tempo, 100% da velocidade do clock é utilizado pelo processador.

Core: O core é o núcleo do processador. Existem processadores core e multicore, ou seja, processadores com um núcleo e com vários núcleos na mesma peça.

Processador

Características do processador:

Cache: A memória Cache é um tipo de memória auxiliar, que faz diminuir o tempo de transmissão de informações entre o processador e outros componentes

Potência: Medida em Watts é a quantia de energia que é consumida.

$$W = \text{TENSÃO} \times \text{CORRENTE} \times \text{TEMPO}$$

$$P = E \times I$$

Processador

Características do processador:

O processador manipula as instruções dos programas que estão em execução no computador.

Uma Instrução é formada pelas seguintes partes:



OPCODE – Código de Operação – informa a AÇÃO da instrução

Operando – São valores ou referência a conteúdos de registros ou variáveis

$$A = 4 \quad B = 5$$

INSTRUÇÃ^O : SIMPLES / NORMAL
N^{ão} É DE DESVIO

INSTRUÇÃ^O : DESVIO DE UMA CONDIÇÃO

DESVIAR O FUXO NORMAL
DAS INSTRUÇÕES PARA
UMA OUTRA POSIÇÃO (ENDEREÇO)
DE MEMÓRIA.

IF 1000
 ELSE
 5000
 ENDIF

RETURN - VOLTAZ - RETOUR

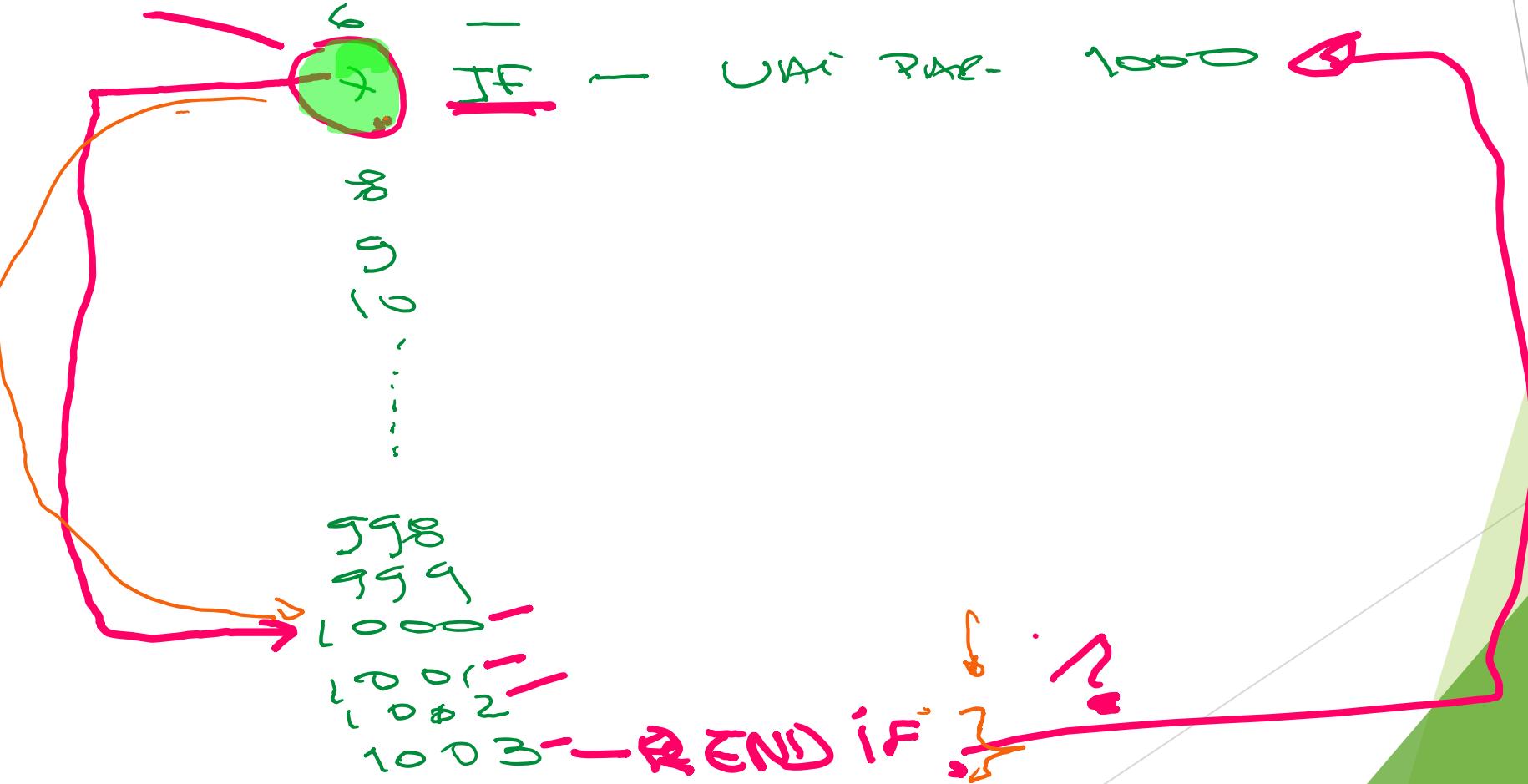
INSTR

1 — NORM
2 — NORM
3 — NORM
4 — NORM
5 —
6 —
7 —
8 —
9 —
10 —

SALVO

IF

— WAIT TIME 1000



```
#include <stdio.h>
#include <stdbool.h>

int main(void) {
    if(true) {
        printf("A instrução é verdadeira!\n");
    }

    return 0;
}
```

Processador

Componentes internos da CPU

RI/UC – Registro de Instrução e Unidade de Controle

ULA – Unidade Lógica Aritmética

ACC – Acumulador

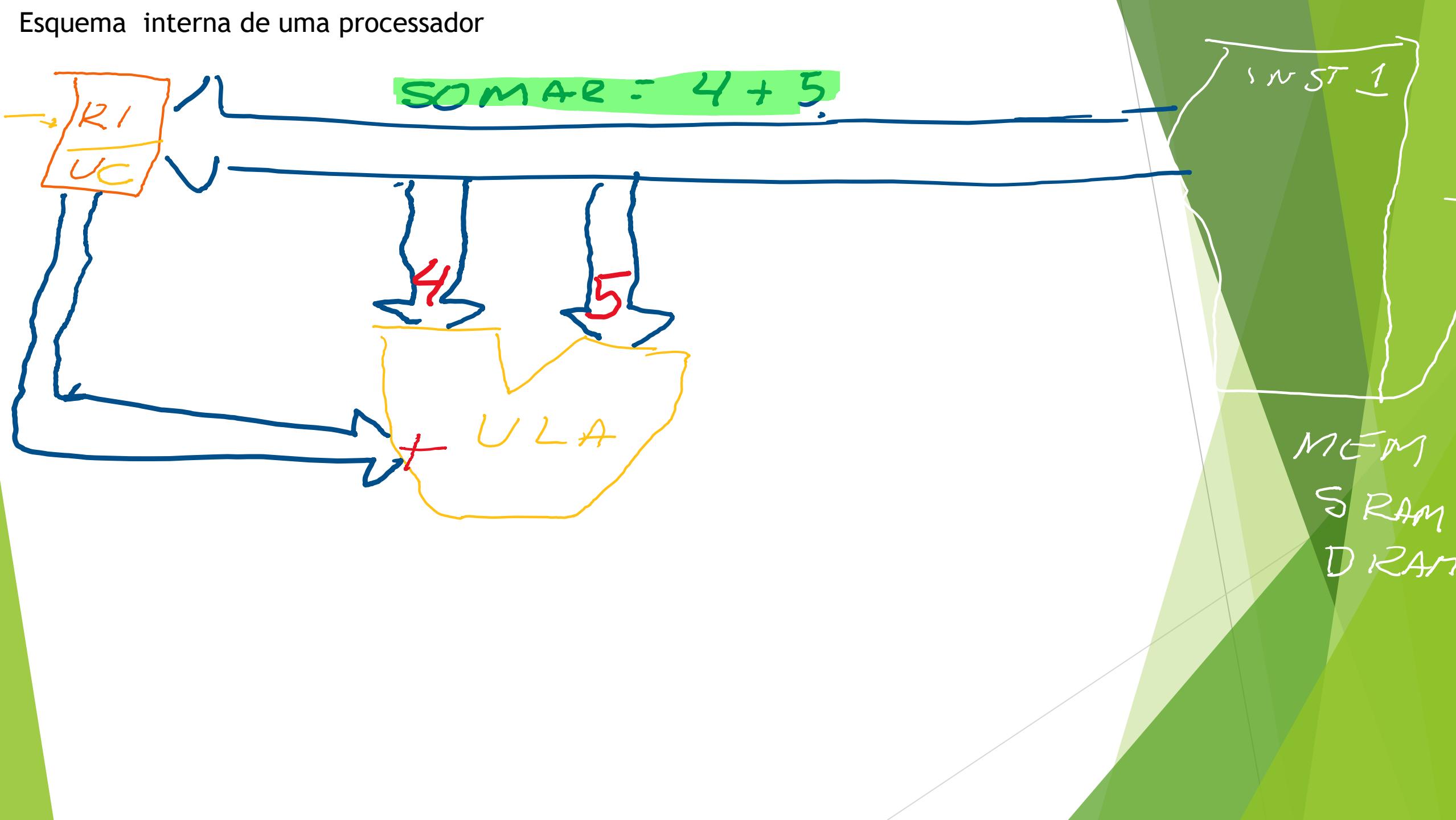
PC – Program counter

SP – Stack pointer

FLAG

Registradores

Esquema interna de uma processador



Processador

Componentes internos da CPU

RI/UC – Registro de Instrução e Unidade de Controle

Busca uma instrução da memória, realiza a sua decodificação (o que esta instrução irá realizar), informa a ULA o tipo de processamento e envia os valores nas entradas da ULA.

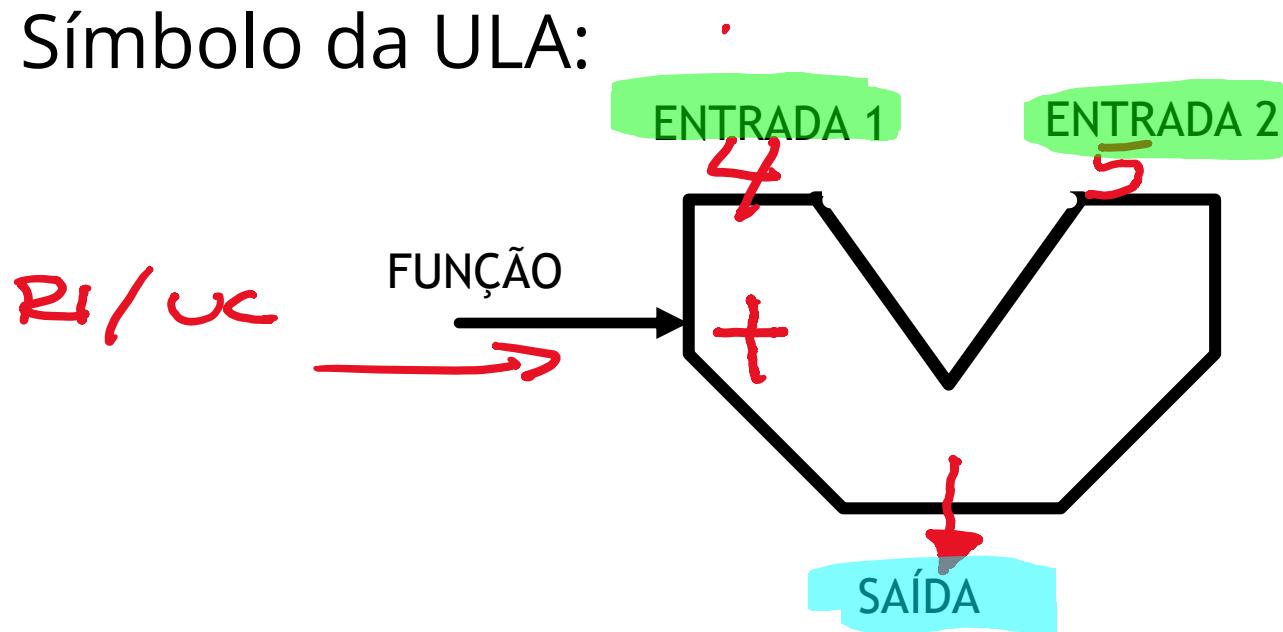
Processador

Componentes internos da CPU

ULA – Recebe do RI/UC qual operação ela irá executar e aplica nos valores de suas entradas. A ULA realiza funções de Lógica binária, aritmética, de complemento, deslocamento etc.

+

Símbolo da ULA:



| | | | | |
|----------------------|----------|----------------|------------------------|-----------------------------|
| LÓGICA BIUÁRIA | AND | - E | = | |
| | OR | - O | = | |
| | NAND | = \bar{O} | - | |
| | NOR | = \bar{E} | - | |
| | NOT | = \bar{O} | - | |
| | XOR | | INVERSOR | - |
| | | | OU EXCLUSIVO | - |
| <hr/> | | | | |
| LÓGICA ARITMÉTICA | + - | LOG | + $\frac{-}{\sqrt{x}}$ | $\times \frac{1}{x^2}$ LOG. |
| | \times | EXPON | | |
| | : | | | |
| <hr/> | | | | |
| COMPLEMENTO | - | INVERTE O BYTE | * | 1. |

DESVOCAMIENTO —
 DESVOCA (SHIFT) DIRECCIÓN
 RIGHT

 ESQUERDA
 LEFT

$$\begin{array}{r}
 01101010 \\
 01001010 \\
 + 1 \\
 \hline
 01001011
 \end{array}$$

$$\begin{array}{r} 12 \rightarrow \text{COMPLEMENTO} + 1 \\ - 8 \\ \hline 4 \end{array}$$

$$\begin{array}{r} 12 \rightarrow 0\overset{1}{1}00 \rightarrow 0\overset{1}{0}\overset{1}{1}1 \\ \quad \quad \quad +1 \\ \hline 0100 \\ 8 \rightarrow \qquad \qquad \qquad 1000 \\ \hline \cancel{1}100 \\ \qquad \qquad \qquad \swarrow 21 \end{array}$$

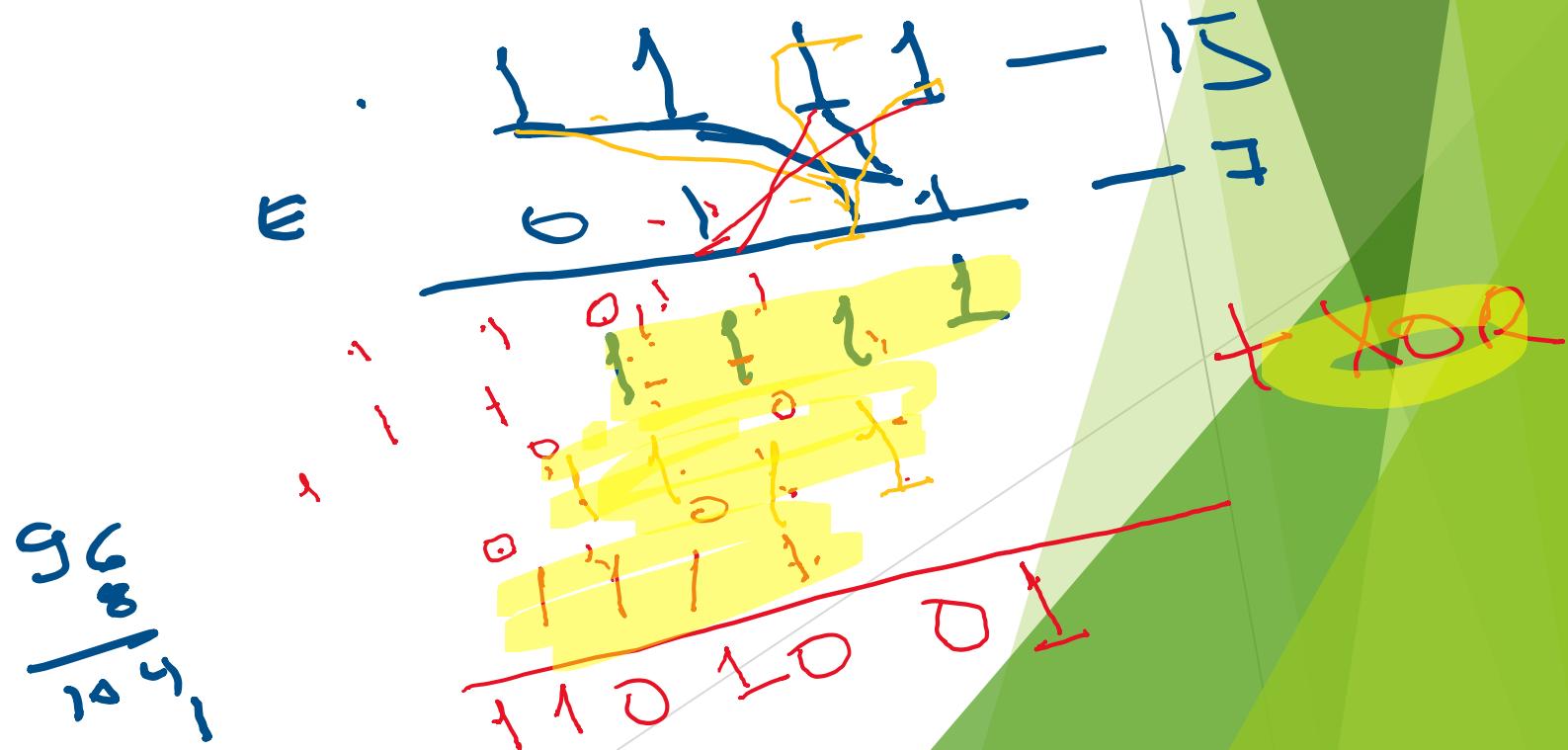
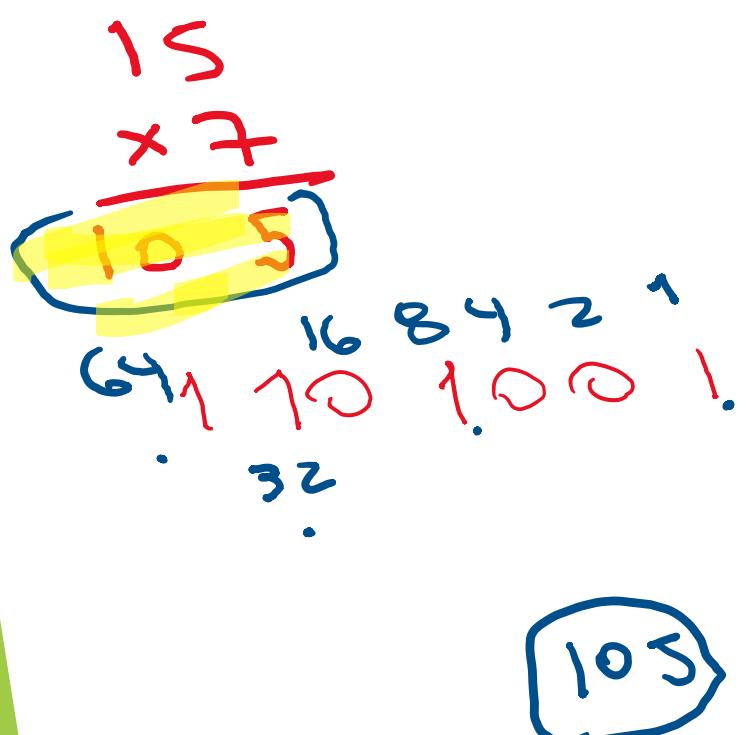
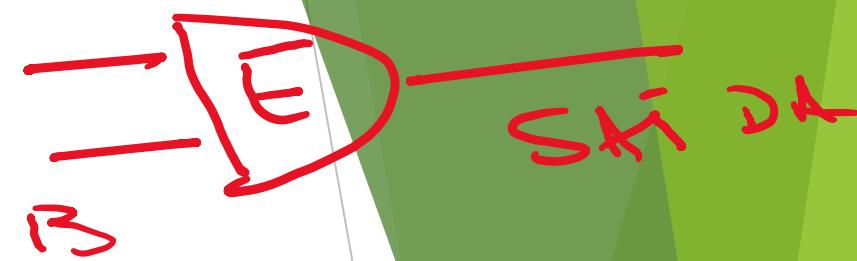
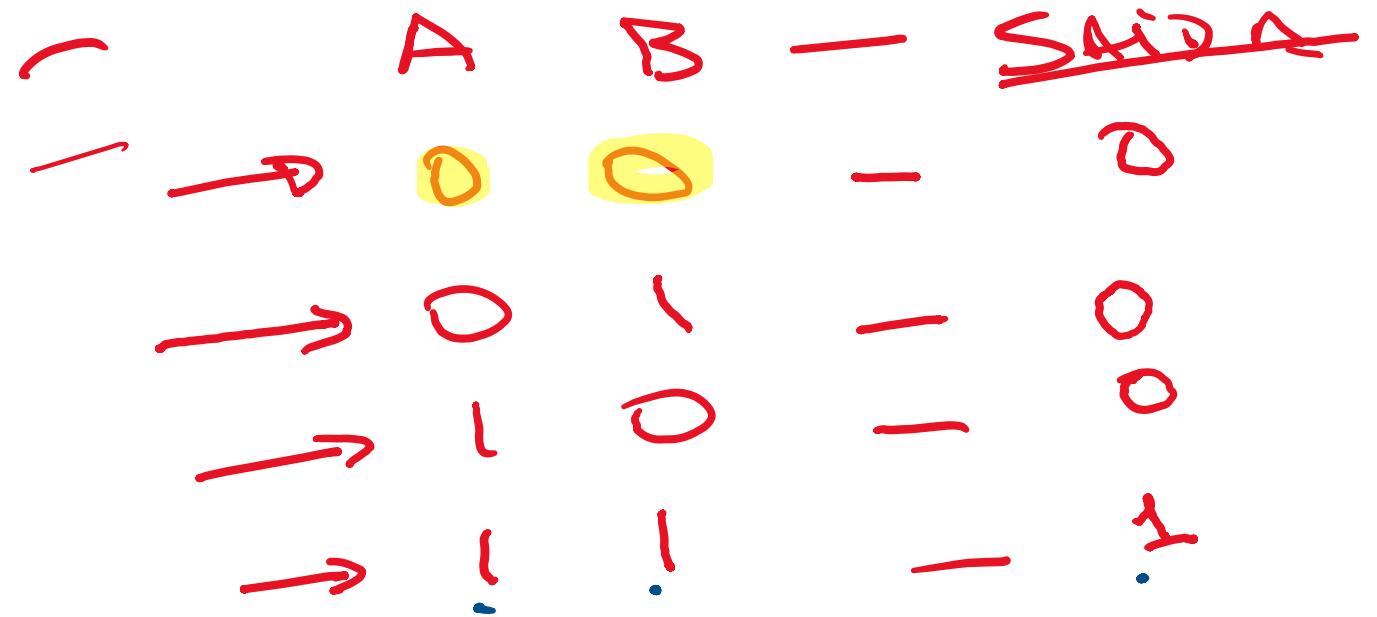
$$0100 = 4$$

$$\begin{array}{r} \text{15} \\ - 7 \\ \hline 8 \end{array} \quad \begin{array}{r} 8 \ 4 \ 2 \ 1 \\ , , , , \rightarrow 0000 \\ 0 \ 1 \ 1 \ 1 \\ + 1 \\ \hline 0001 \\ 0 \ 1 \ 1 \ 1 \\ + \\ \hline 1000 = 8 \end{array}$$

8421
, , , , → 0000
0111
+ 1

0001
0111
+

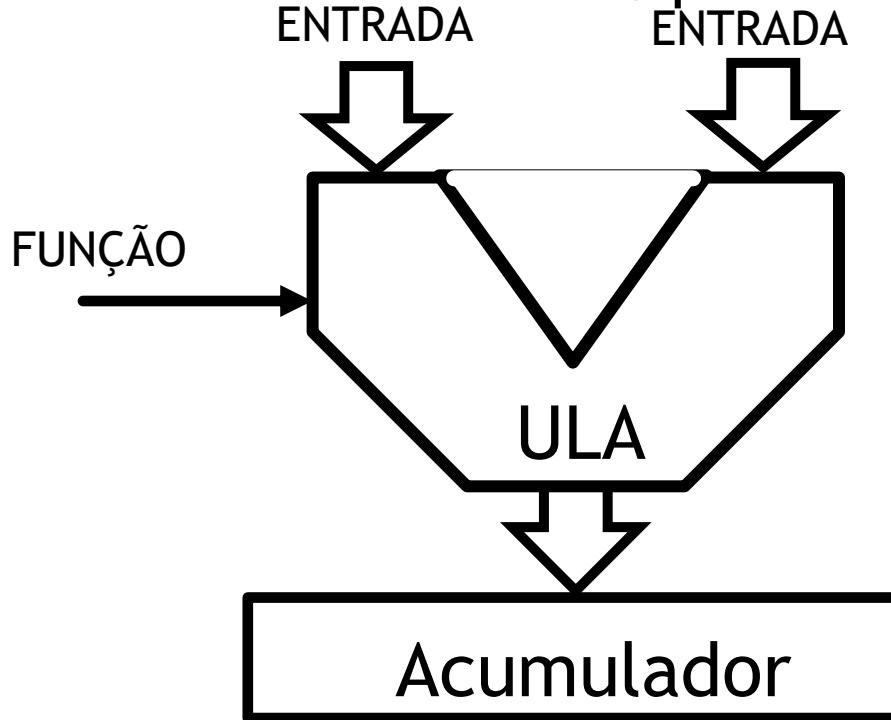
1000 = 8



Processador

Componentes internos da CPU

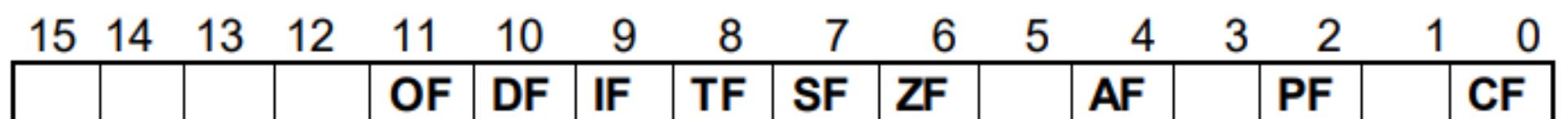
ACC – Acumulador - Todo processamento da ULA será armazenada no Acumulador, pois está fisicamente conectada.



Processador

Registro Flag:

Este registro é utilizado como um sinalizador de todas as operações que a ULA realizar. Este registro é tratado bit a bit a cada operação, a cada instrução que a ULA executar. Cada bit tem seu significado e vários destes bits poderão ser atualizados na execução de uma instrução. O Sistema Operacional e os aplicativos, a cada execução da instrução analisa os valores destes bits.



Alguns destes bits :

Processador

Registro Flag:

- **ZF (Zero Flag):** indica se o resultado de uma operação aritmética é igual a zero (1) ou diferente de zero (0).
- **SF (Sign Flag):** indica se o resultado de uma operação com sinal é positivo (0), se for negativo (1).
- **IF (Interrupt Flag):** indica se as interrupções estão habilitadas (1) ou não (0).
- **DF (Direction Flag):** para operações com strings, se D=0 os registradores de índice serão incrementados, caso contrário (D=1) serão decrementados.
- **OF (Overflow Flag):** indica um estouro da capacidade de armazenamento de um registrador. Se o bit for um (1) significa estouro da capacidade, caso contrário o valor será zero (0).
- **CF (Carry Flag):** indica se houve o ‘vai um’ no cálculo que a ULA executou, se houve logo este bit tem valor 1, sena valaor zero (0)



Processador

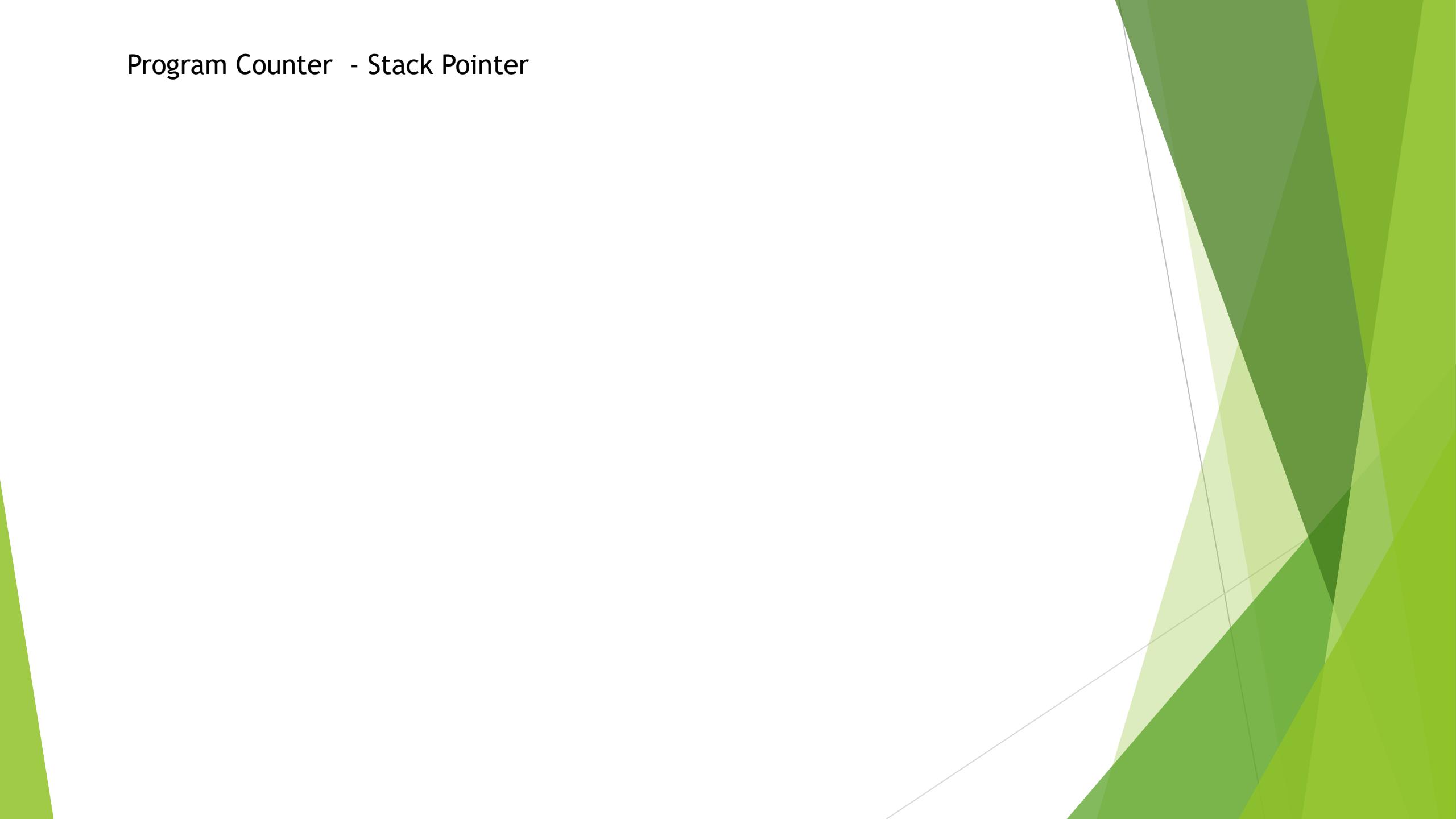
Componentes internos da CPU

PC – Program counter – Contador de Programa

Um contador de programa (PC) é um registro que possui o **endereço da próxima instrução** a ser executada a partir da memória. É um contador necessário para uma execução mais rápida de tarefas, bem como para rastrear o ponto de execução atual.

O **RI/UC** ao decodificar uma instrução irá saber que a próxima instrução será a próxima da sequência, logo atualiza o PC. Se a instrução for, por exemplo, um desvio (IF), o registro SP – stack point será utilizado

Program Counter - Stack Pointer



Processador

Componentes internos da CPU

SP – Stack Pointer – Ponteiro de pilha

É um pequeno registro que armazena o endereço da última solicitação de programa em uma pilha durante uma execução de um desvio . Uma pilha é um buffer especializado que armazena dados de cima para baixo. À medida que novos pedidos chegam, eles "pressionam" os mais antigos, do tipo de último a entrar / primeiro a sair.

Processador

Componentes internos da CPU

SP – Stack Pointer / PC – Program Counter

Em condições normais de uma instrução sem desvio o PC fica sempre apontando a próxima instrução da sequência, porém quando houver um desvio, o próximo endereço será um diferente desta sequência, ou para frente ou para trás de onde estava. Só que ao retornar o programa deverá voltar no endereço de onde estava. É neste ponto que entra as funcionalidades do SP - Stack Pointer...

Processador

Componentes internos da CPU

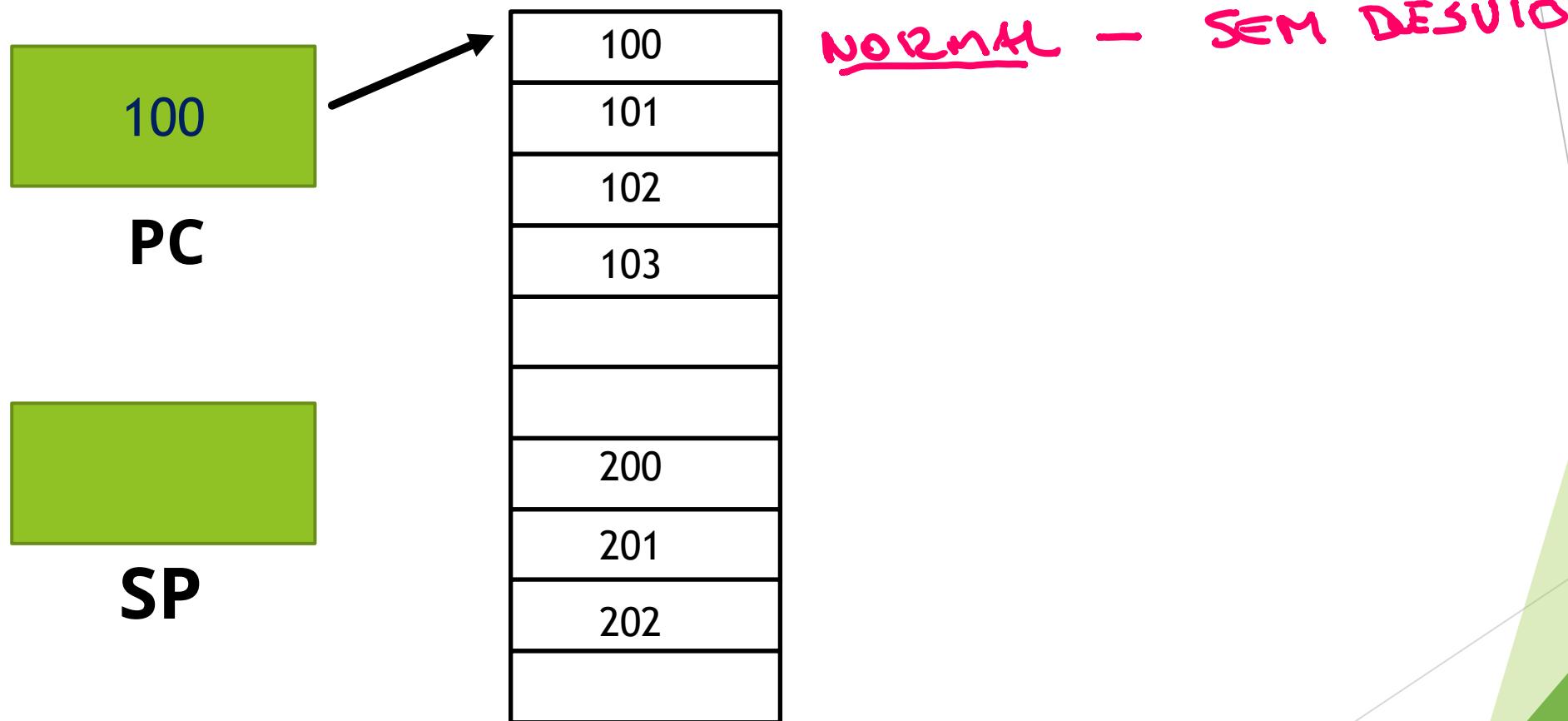
SP – Stack Pointer / **PC** – Program Counter

Logo quando tiver um desvio para outro endereço , o endereço atual do PC é salvo automaticamente no SP e o PC recebe o endereço do desvio e o programa começa a ser executado nesta nova posição do desvio.

Quando terminar esta rotina do desvio (retorno), o programa recupera o endereço do PC (endereço antes do desvio) como não houvesse acontecido nada.

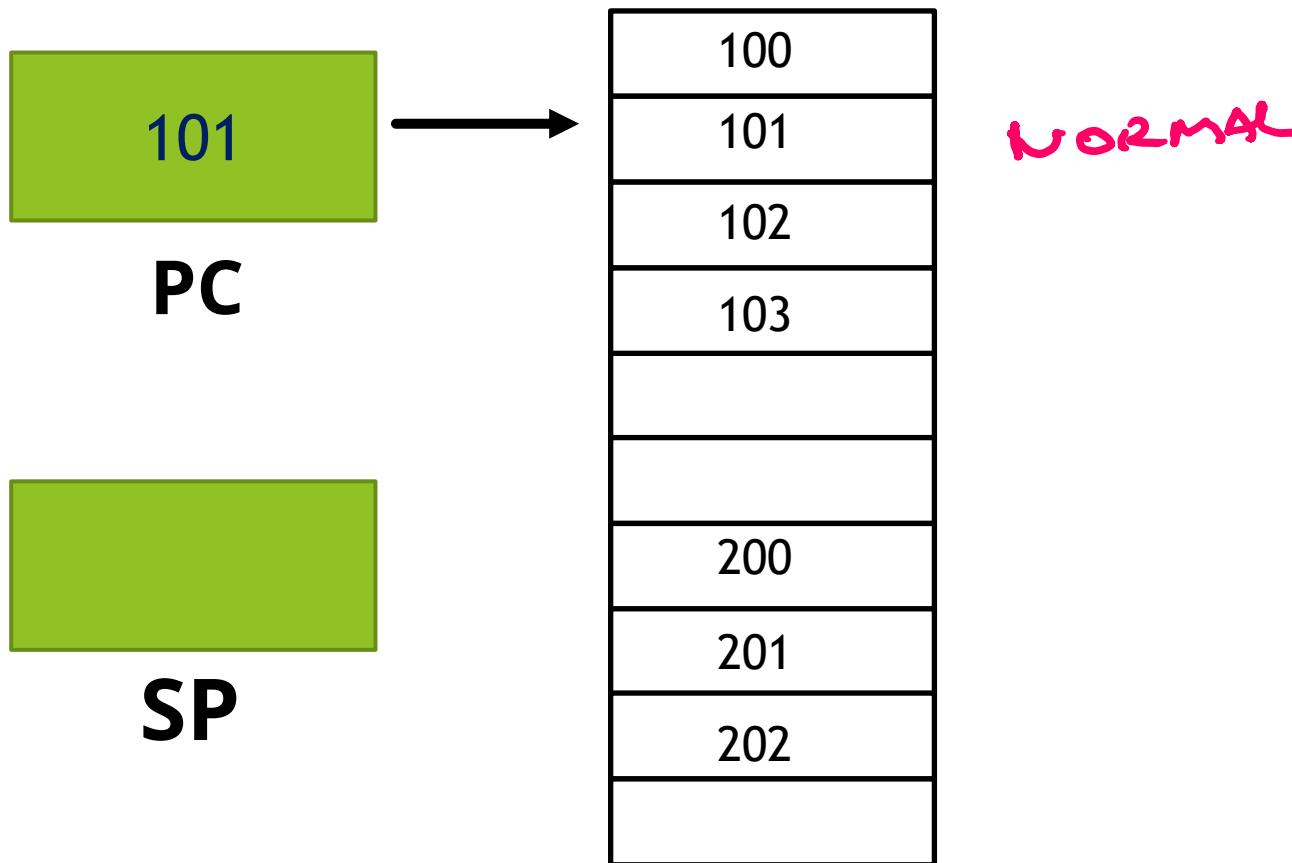
Processador

SP - Stack Pointer / **PC** - Program Counter



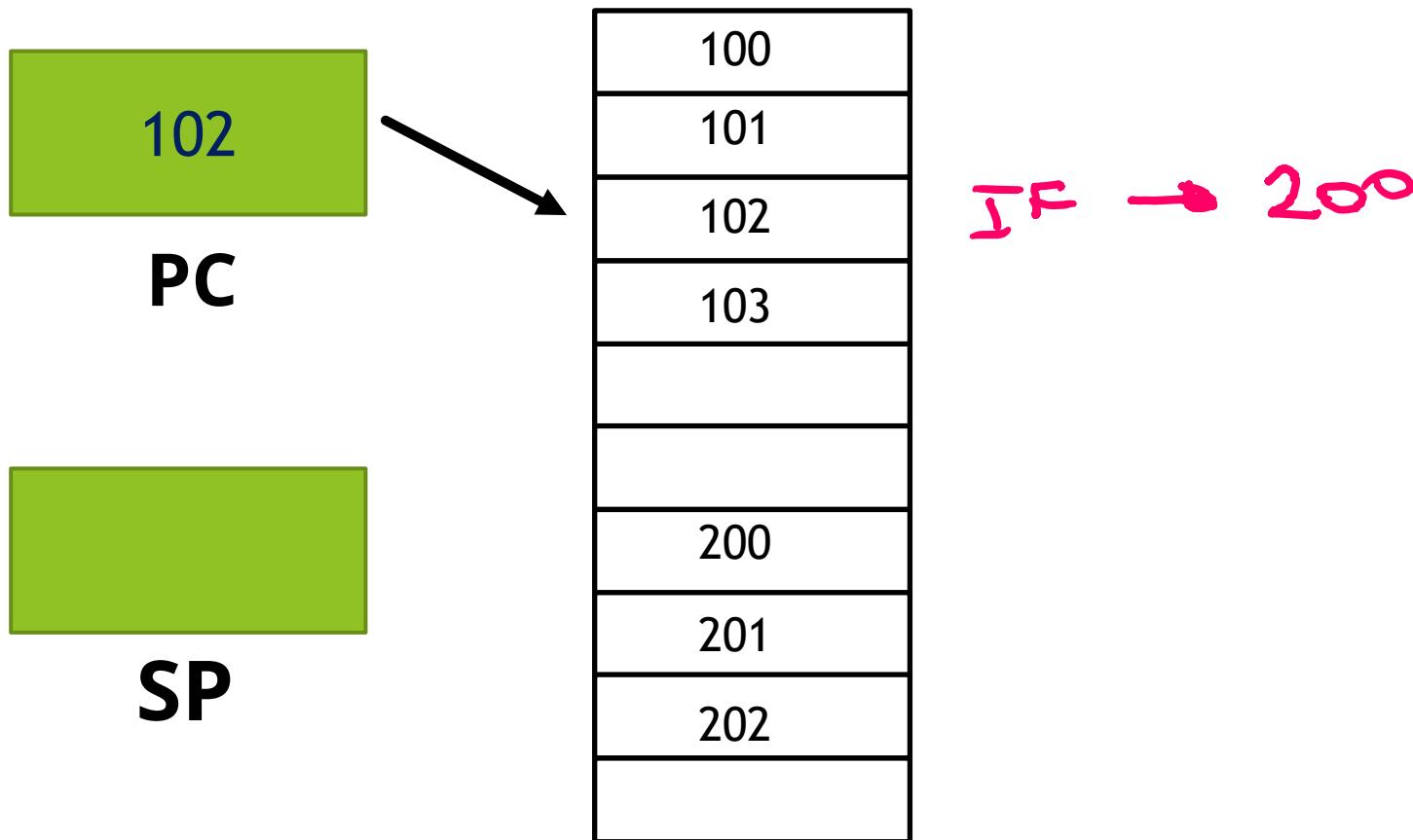
Processador

SP - Stack Pointer / **PC** - Program Counter



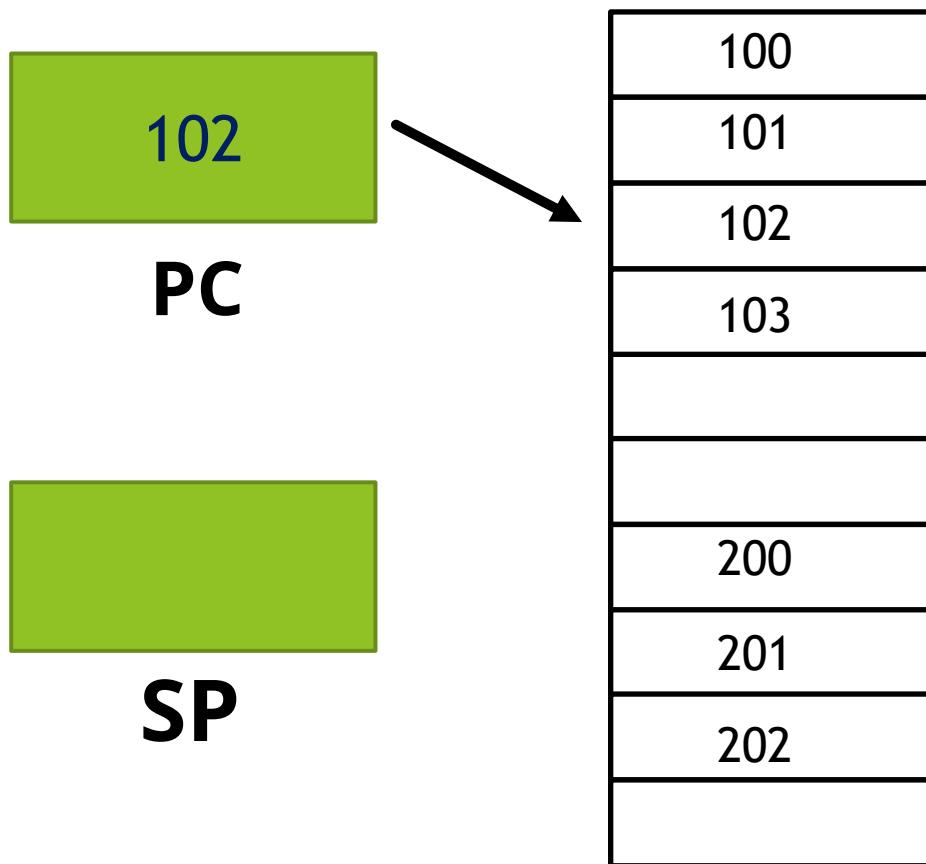
Processador

SP - Stack Pointer / **PC** - Program Counter



Processador

SP – Stack Pointer / **PC** – Program Counter

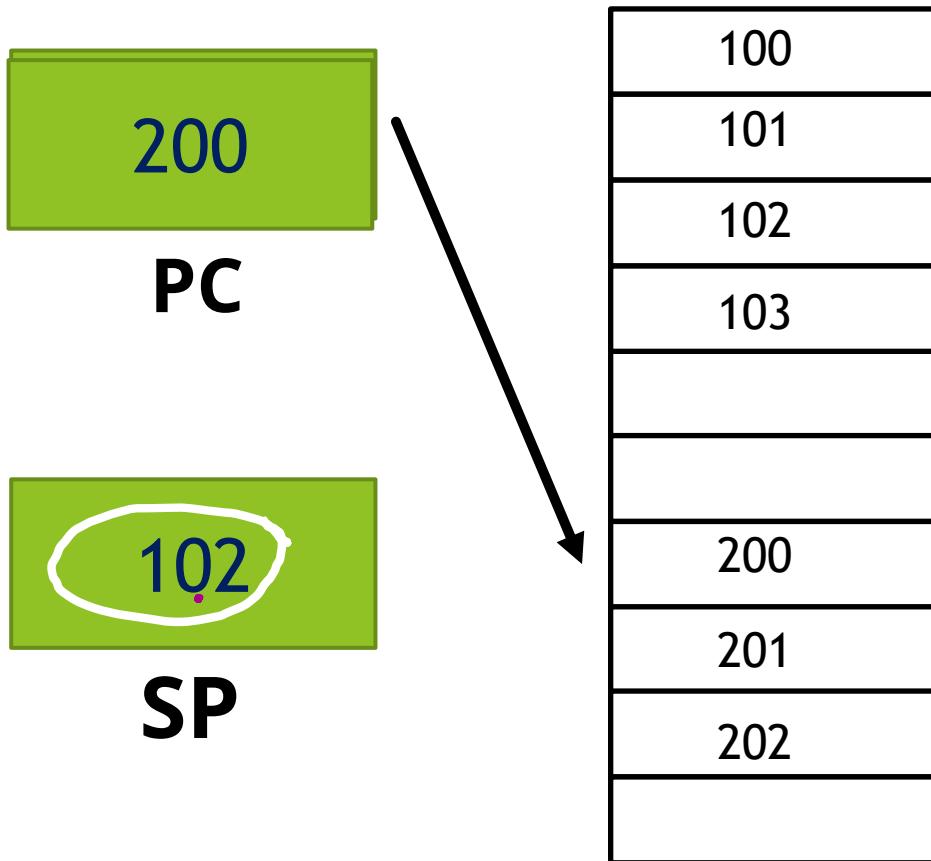


Neste instante chega uma instrução de desvio que o RI/UC decodificou e o próximo endereço é o 200.

Logo o 200 irá para o PC mas antes temos que salvar o endereço atual do PC.

Processador

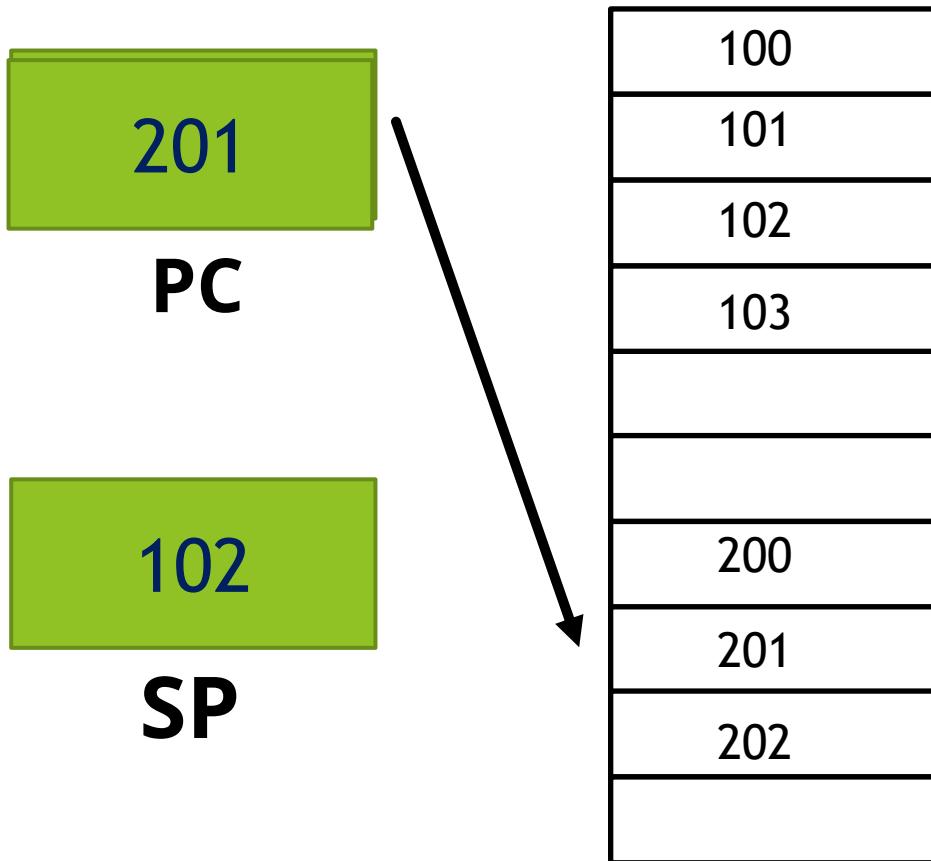
SP – Stack Pointer / **PC** – Program Counter



Então o 102 irá para o Stack Pointer SP e o novo endereço 200 para o PC

Processador

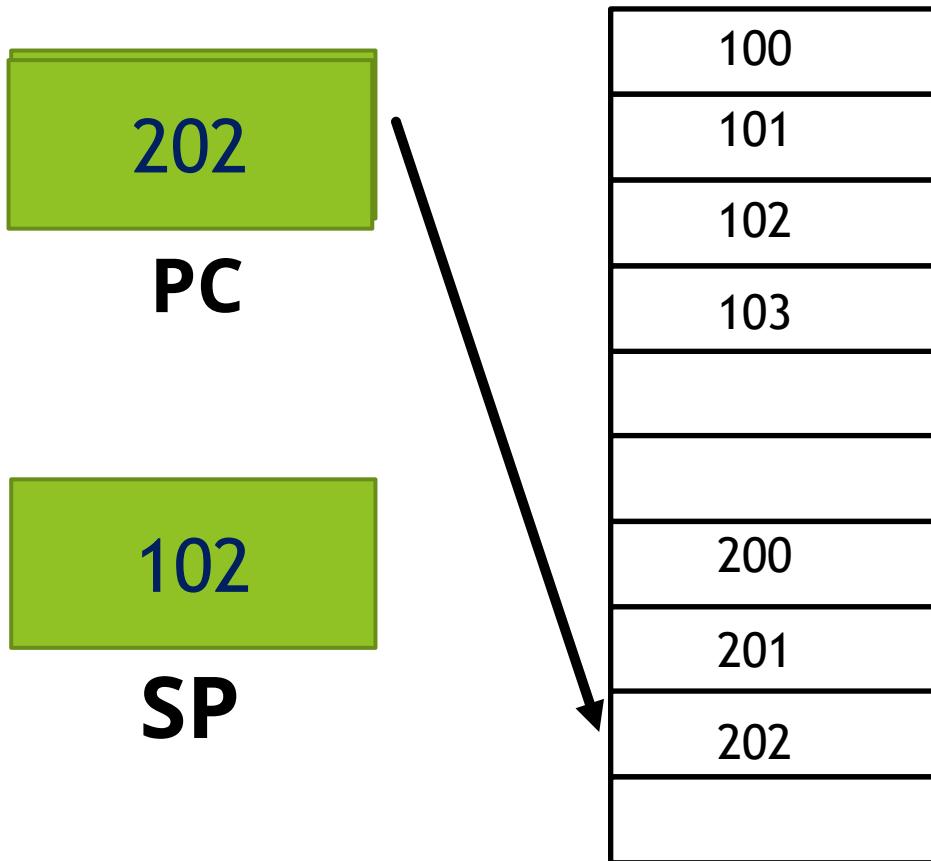
SP – Stack Pointer / **PC** – Program Counter



PC recebe o endereço da próxima instrução, ou seja 201

Processador

SP – Stack Pointer / **PC** – Program Counter



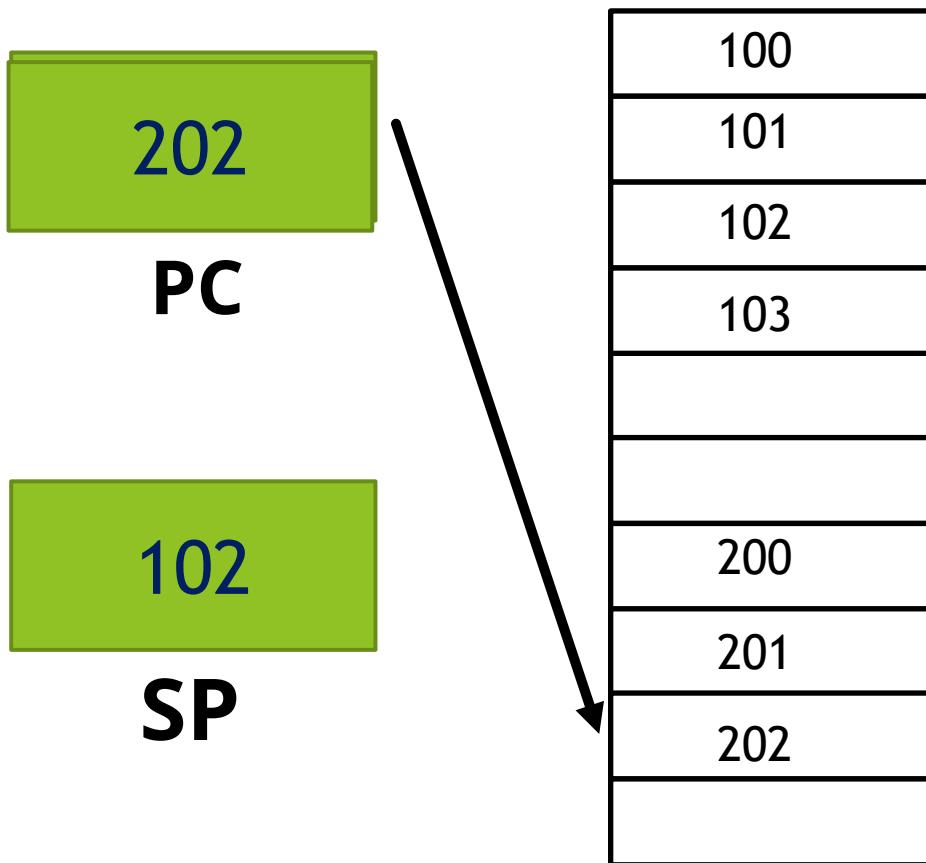
PC recebe o endereço da próxima instrução, ou seja 202. Vamos supor que neste endereço tem uma instrução de retorno (return).

RETURN - ENDIF

R/UC -

Processador

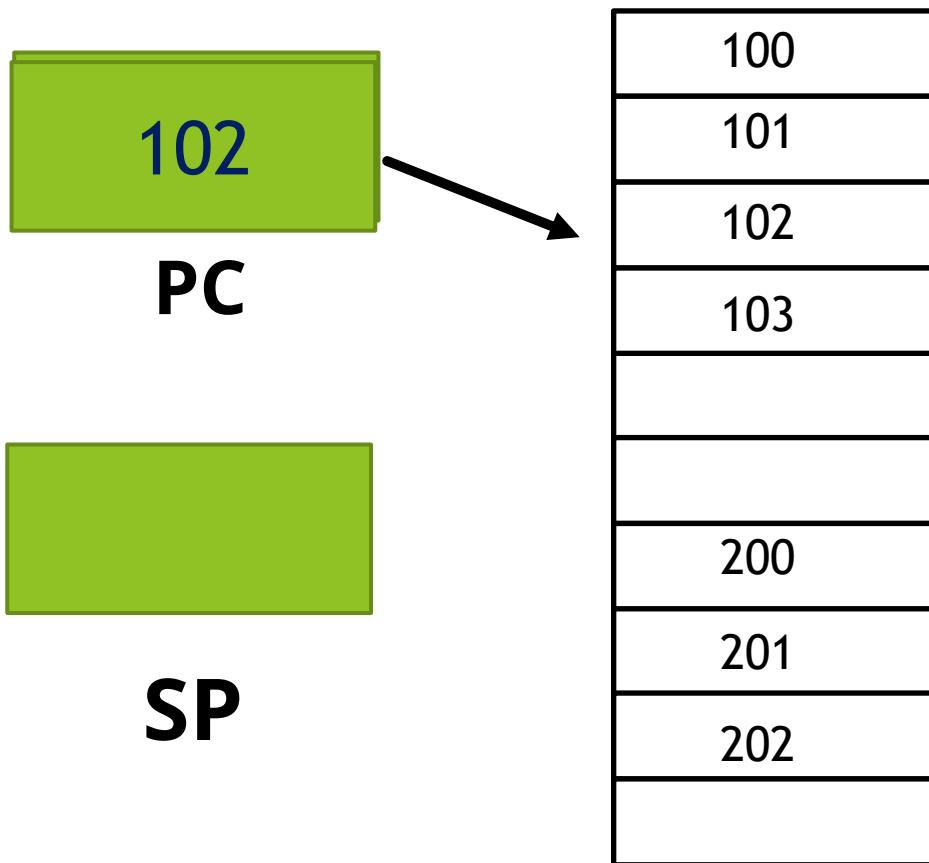
SP – Stack Pointer / PC – Program Counter



Neste retorno o gerenciador observa o valor do SP-Stack Pointer e devolve para o PC-Program Counter, ou seja o PC irá receber o 102.

Processador

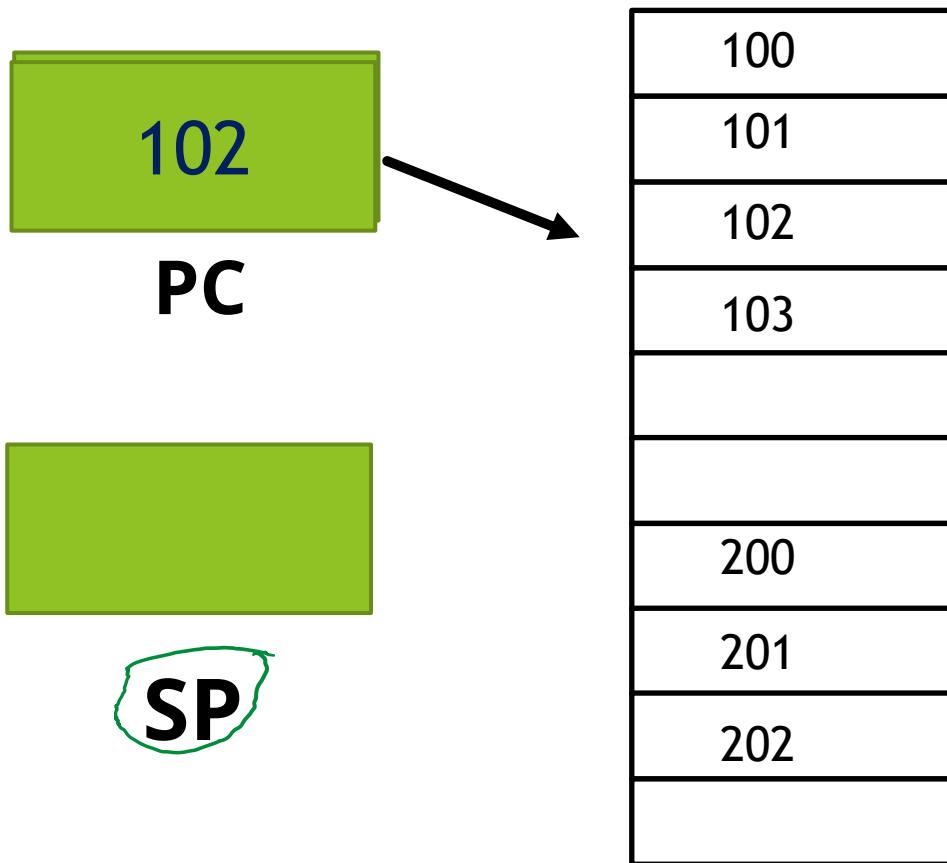
SP – Stack Pointer / **PC** – Program Counter



Neste retorno o gerenciador observa o valor do SP-stack Pointer e devolve para o PC-Program Counter, ou seja o PC irá receber o 102.

Processador

SP – Stack Pointer / PC – Program Counter



Voltando como era antes
do desvio.

Processador

Componentes internos da CPU

Registradores: Pequeníssimas áreas de memória temporária (buffer), de tamanho de 32 ou 64 bits, apoiando o processador, alguns deles :

- *AX*
- *BX*
- *CX*
- *DX*
- *FLAG*

Processador

AX: é o chamado **ACUMULADOR** (o X refere-se a “eXtended”). **Todo** resultado dos cálculos da ALU irá SEMPRE ser armazenado neste registro, pois ele está conectado fisicamente na saída da ULA.

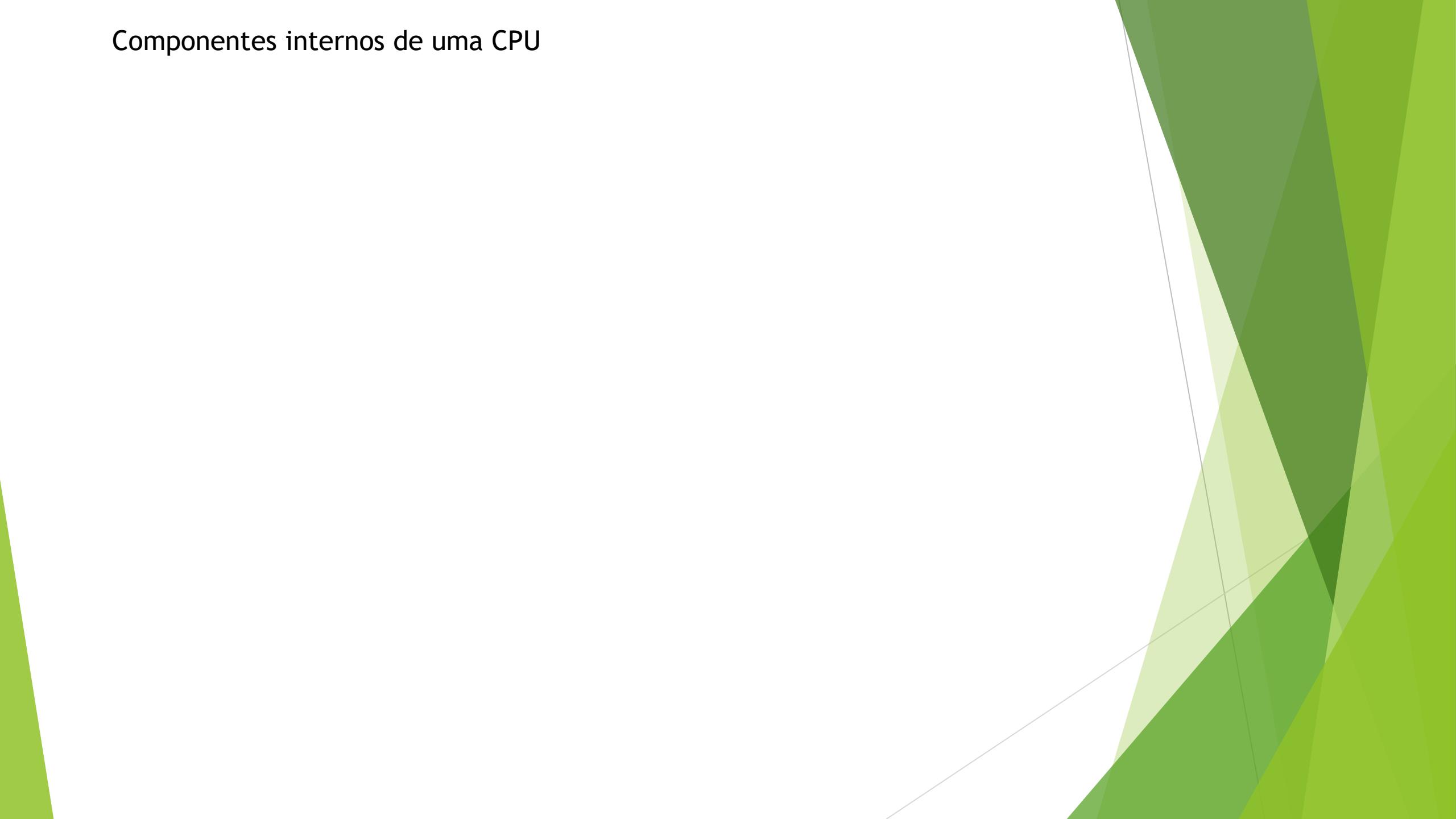
BX: É chamado registrador de **BASE**, pois pode ser usado como base de endereçamento para manipulação apoio ao registrador AX. Na maioria das instruções os registros BX e AX estão presentes

Processador

CX: É chamado registrador de **CONTAGEM**, pois além de trabalhar com dados também é usado para contagens em instruções com “LOOPS” e instruções de desvios (condições).

DX: É chamado de registrador de **DADOS**, pois normalmente este registro utiliza strings de caracteres (sequência de caracteres)

Componentes internos de uma CPU



LOD = Load

LOD X = carregar o valor do Registro x

STO – STORE - Armazena

SUB – Subtração

HLT – HALT → Parar o processamento

Processador

Exemplos de manipulação de registradores:
Linguagem Assembly

Uma Instrução é formada pelas seguintes partes:

| | | |
|--------|------------|------------|
| OPCODE | Operando 1 | Operando 2 |
|--------|------------|------------|

OPCODE : é o Código de Operações, ou seja uma ação que a instrução irá executar , exemplo:

Processador

ADD - adição

SUB - subtração

MUL – multiplicação

DIV - divisão

MOV - transferir um valor ou conteúdo de um registro

Em Assembly iremos **SEMPRE TRABALHAR COM NÚMEROS EM HEXADECIMAIS.**

Processador

Exemplos:

Passar para assembly a equação: $20 + 12$

1) Lembre-se que o resultado SEMPRE cairá no AX (acumulador), primeiramente temos que passar estes valores para Hexadecimal:

$$20_{(10)} \rightarrow \text{14}_{(16)} \quad 12_{(10)} \rightarrow \text{0C}_{(16)}$$

Processador

2) Mover estes valores para os registros AX e BX:
Supondo valores iniciais para AX e BX = 00

| | AX | BC |
|------------|----|----|
| | 00 | 00 |
| MOV AX, 14 | 14 | 00 |
| MOV BX, 0C | 14 | 0C |
| ADD AX, BX | 20 | 0C |

**ATENÇÃO O RESULTADO SEMPRE CAIRÁ DENTRO DO AX,
ELIMINANDO O VALOR QUE ANTES ESTAVA ARMAZERADO.**

Processador

PIPPIN Simulator:

https://www.youtube.com/watch?v=SBX9pma_e-g

Processador

OBRIGADO!

DÚVIDAS.