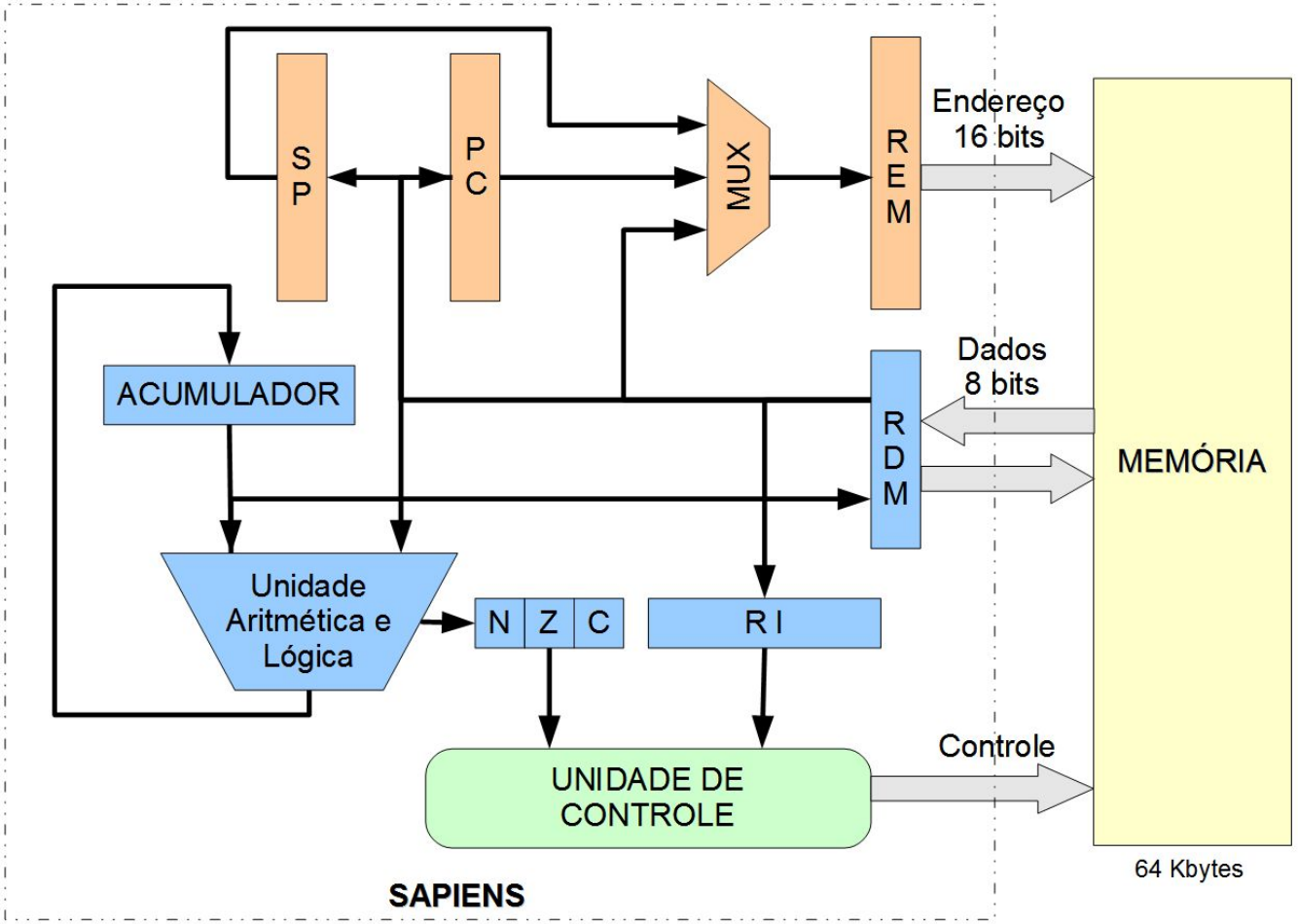
INSTITUTO FEDERAL
Santa Catarina

CST Análise e Desenvolvimento de Sistemas
AOC786201 - Fundamentos de Arquitetura e Organização de Computadores

Introdução à Arquitetura e Organização de Computadores

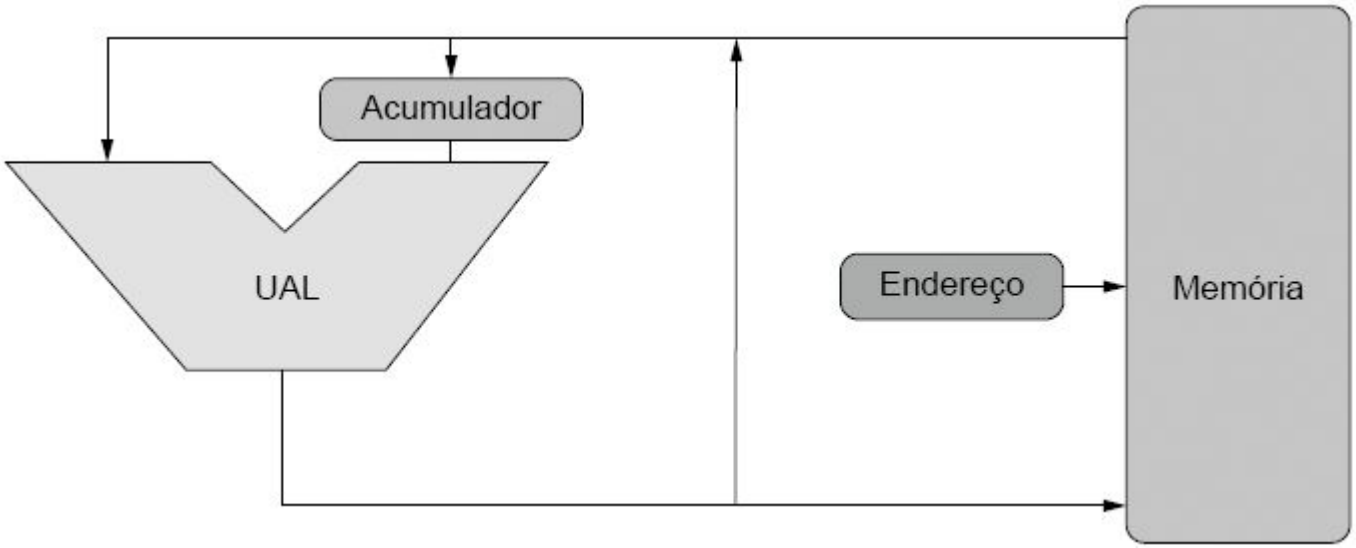
Estudo da arquitetura didática do Processador Sapiens



SILVA, Gabriel Pereira da; BORGES, José Antonio dos S. **Arquitetura e Organização de Computadores – Uma Introdução**. Rio de Janeiro: LTC, 2024. *E-book.* p.132. ISBN 9788521638667. Disponível em: https://app.minhabiblioteca.com.br/reader/books/9788521638667/

Processador Sapiens: Acumulador

- Uma arquitetura de 8 bits, com um acumulador também de 8 bits
  - O acumulador é um registrador especial colocado junto à unidade aritmética e lógica (UAL) com o intuito de agilizar as operações realizadas pelo processador.



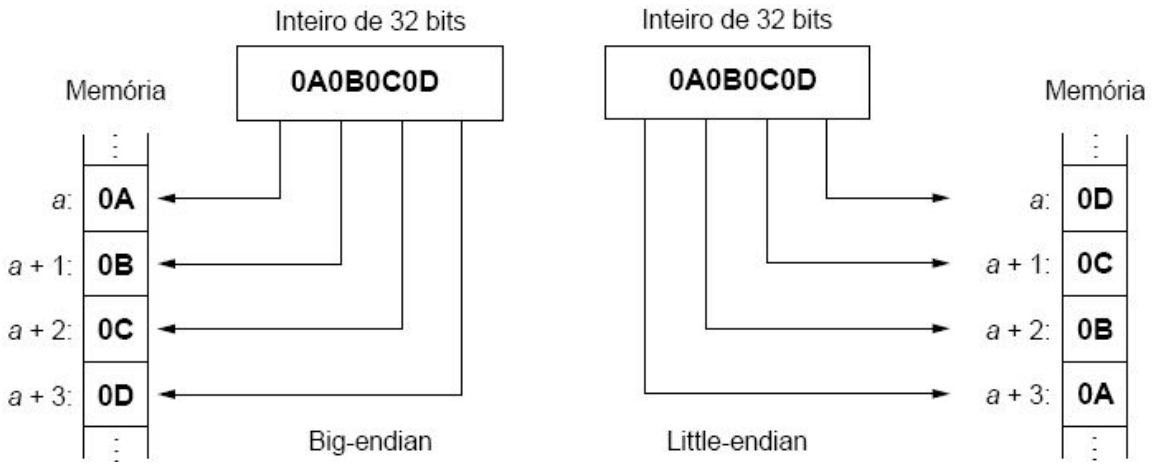
- Além da velocidade de acesso ao acumulador ser superior à velocidade de acesso de dados armazenados na memória, o acumulador também serve para armazenamento de dados parciais de operações mais complexas.

Processador Sapiens: Instruções

- Instruções de 8 bits, com até 2 bytes como parâmetros adicionais
  - As instruções em linguagem de máquina do processador Sapiens podem ter um, dois ou três bytes
  - 6 bits são utilizados para o OpCode da instrução
  - 2 bits para o modo de endereçamento de memória
  - 8 + 8 bits para endereços

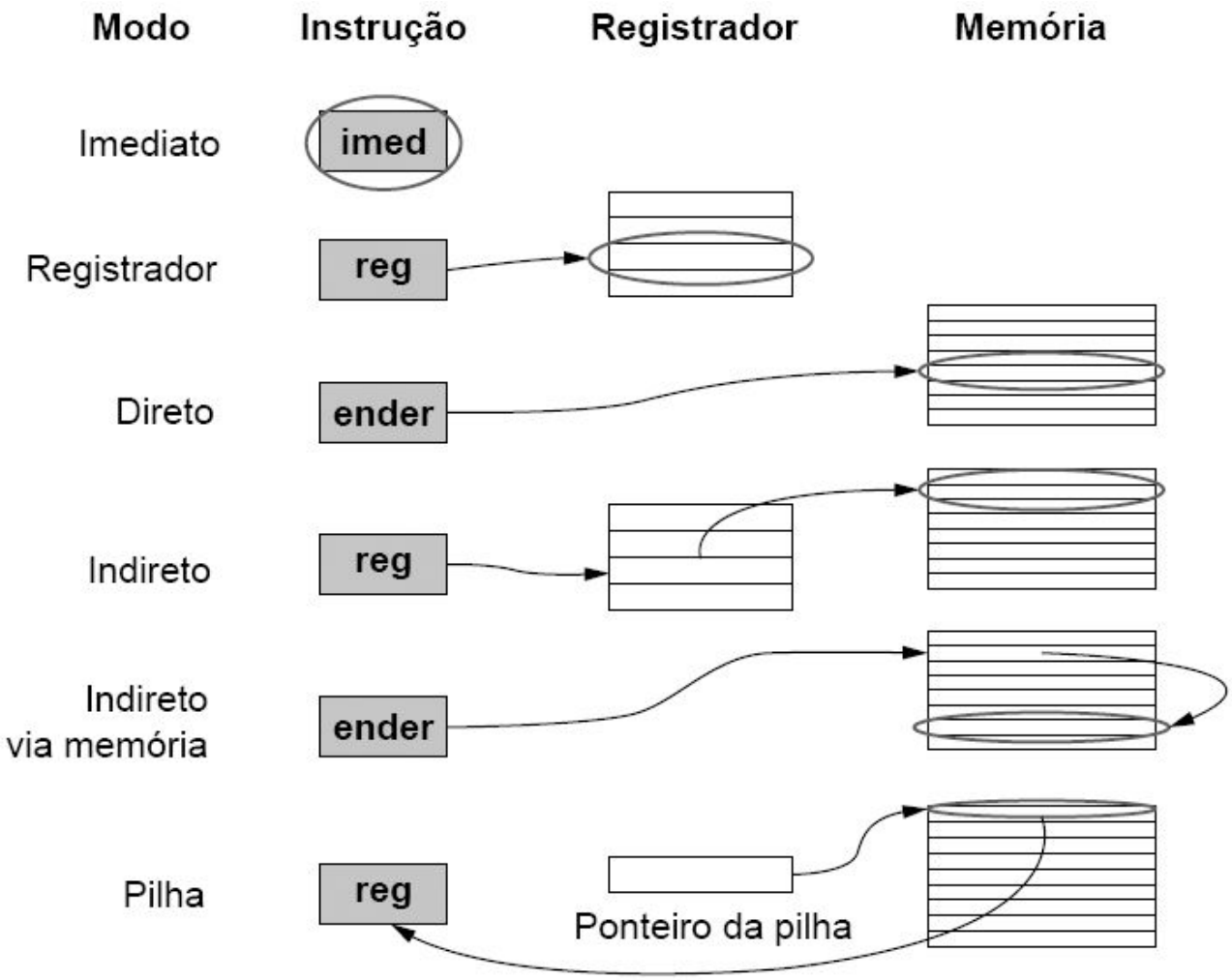
Processador Sapiens: Instruções

- Um apontador de instruções (PC – Program Counter) com 16 bits de largura (little-endian), permitindo endereçar uma memória de 64 kBytes;
  - A ordenação big-endian (BE) armazena o byte mais significativo do operando no menor endereço de memória
  - A ordenação little-endian (LE) armazena o byte menos significativo no menor endereço de memória.
  - Ambos os tipos de ordenação são amplamente utilizados pelos processadores comerciais, LE prevalescendo em arquiteturas x86, implementações antigas ARM e RISC-V. O BE é utilizado em Motorola 68K, nas versões antigas do SPARC e PowerPC, e na maioria dos protocolos de rede.



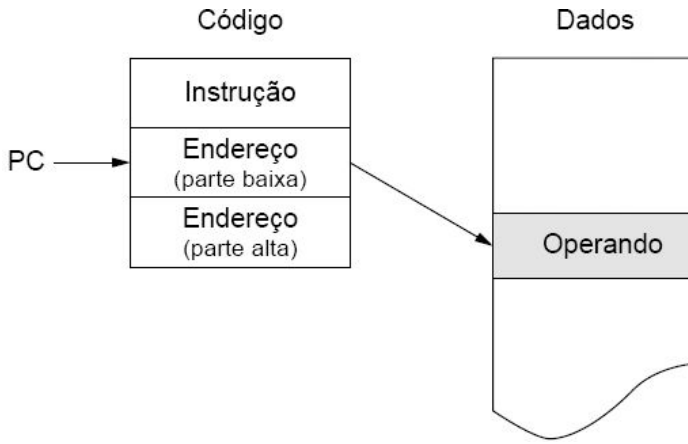
INSTITUTO FEDERAL
Santa Catarina

Modos de endereçamento típicos

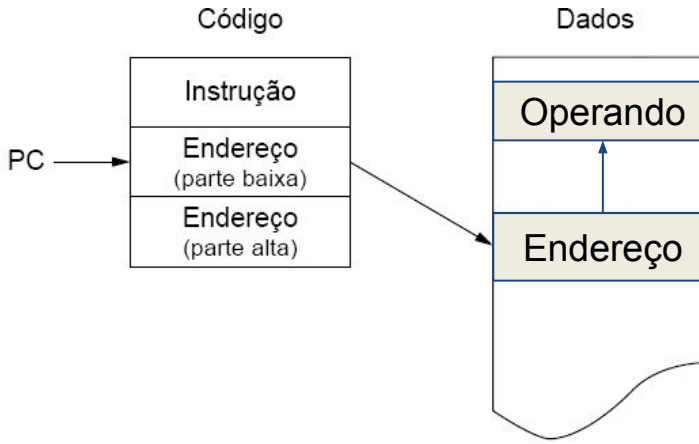


Processador Sapiens: Modos de endereçamento

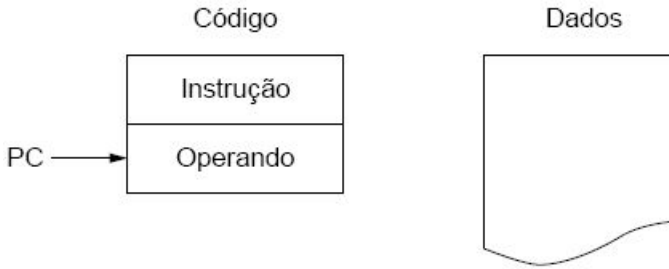
- 00 – Direto: o segundo e terceiro bytes da instrução contêm o endereço do operando na memória



- 01 – Indireto: o segundo e terceiro bytes da instrução contêm o endereço da posição de memória com o endereço do operando (ou seja, é o endereço do ponteiro para o operando). Na linguagem: @ (arrôba)

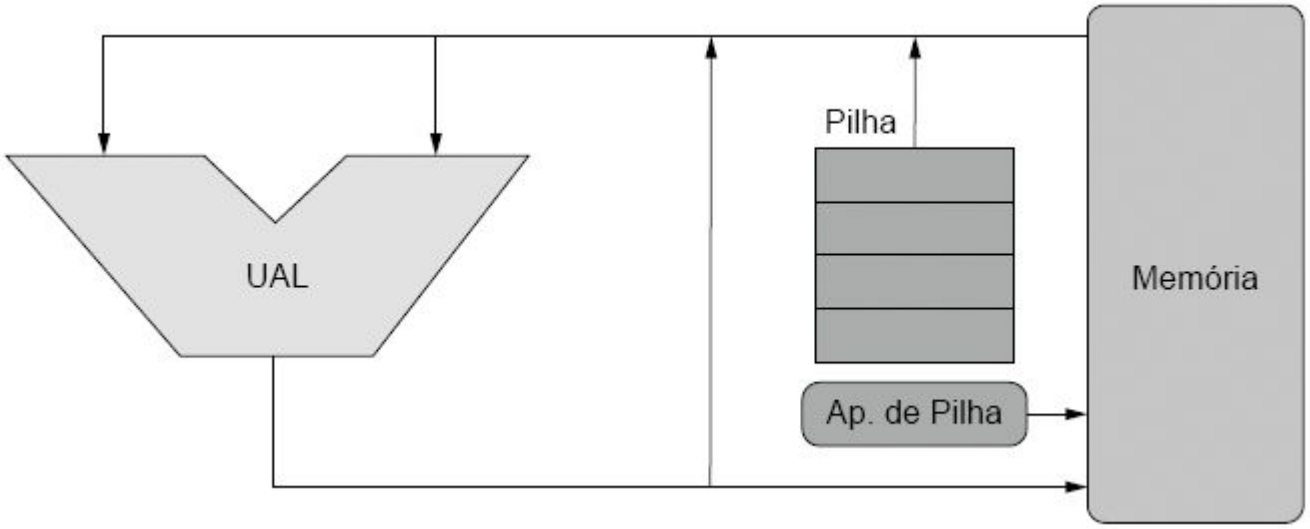


- 10 – Imediato 8 bits: o segundo byte da instrução é o próprio operando. Na linguagem: # (tralha).
- 11 – Imediato 16 bits: os dois bytes seguintes à instrução são utilizados como operando. Na linguagem: Instrução LDS (Load Stack Pointer) com # (tralha)



Processador Sapiens: Pilha

- Um apontador de pilha (SP – Stack Pointer), também de 16 bits, para possibilitar o uso de uma pilha para a chamada e o retorno de rotinas e procedimentos, além da passagem de parâmetros
  - Possui as operações “POP” e “PUSH” para retirar e colocar operandos no topo da pilha. Normalmente os primeiros elementos no topo da pilha se encontram em registradores junto ao processador, e o restante é distribuído na memória a partir do endereço no apontador de pilha.



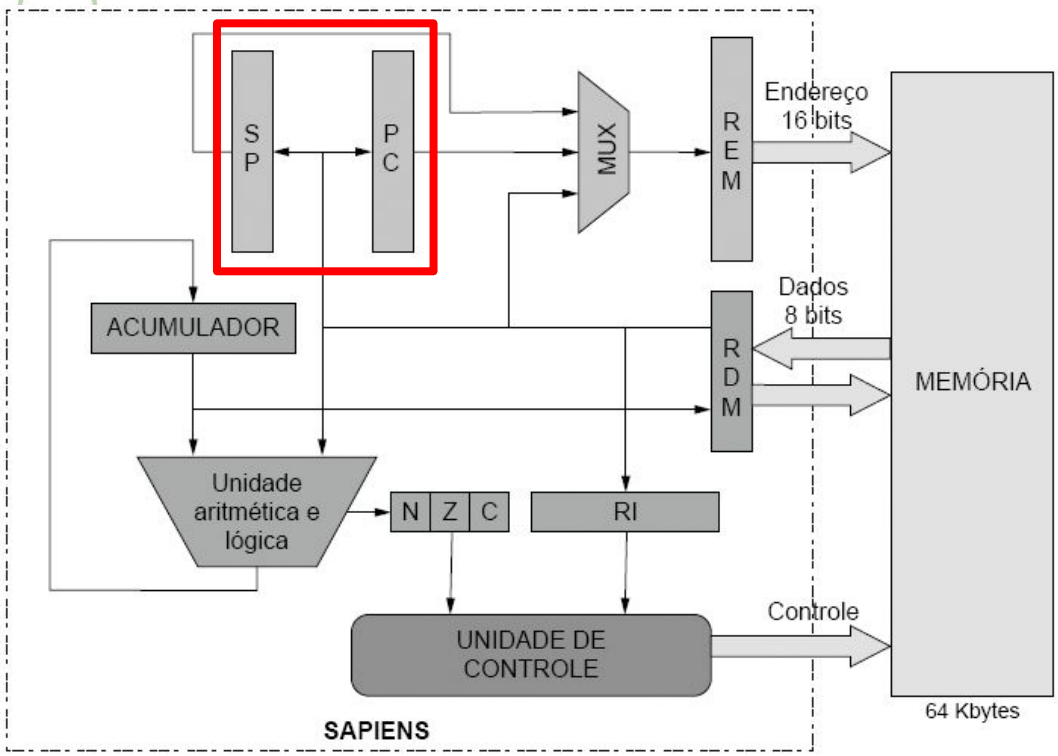
Processador Sapiens: Condições especiais da ULA

- Códigos de condição para indicar o resultado da última operação na UAL:
  - Um código de condição (flag) C (carry) para indicar se houve vai um ou vem um, conforme a operação anterior
    - 1 – o resultado deu vai um ou vem um.
    - 0 – o resultado não deu nem vai um ou vem um.
  - Um código de condição (flag) Z para indicar se o resultado da última operação da UAL foi igual a zero;
    - 1 – o resultado é igual a zero
    - 0 – o resultado diferente de zero
  - Um código de condição (flag) N para indicar se o resultado da última operação UAL foi negativo.
    - 1 – o resultado é negativo
    - 0 – o resultado não é negativo

Processador Sapiens: Condições especiais da ULA

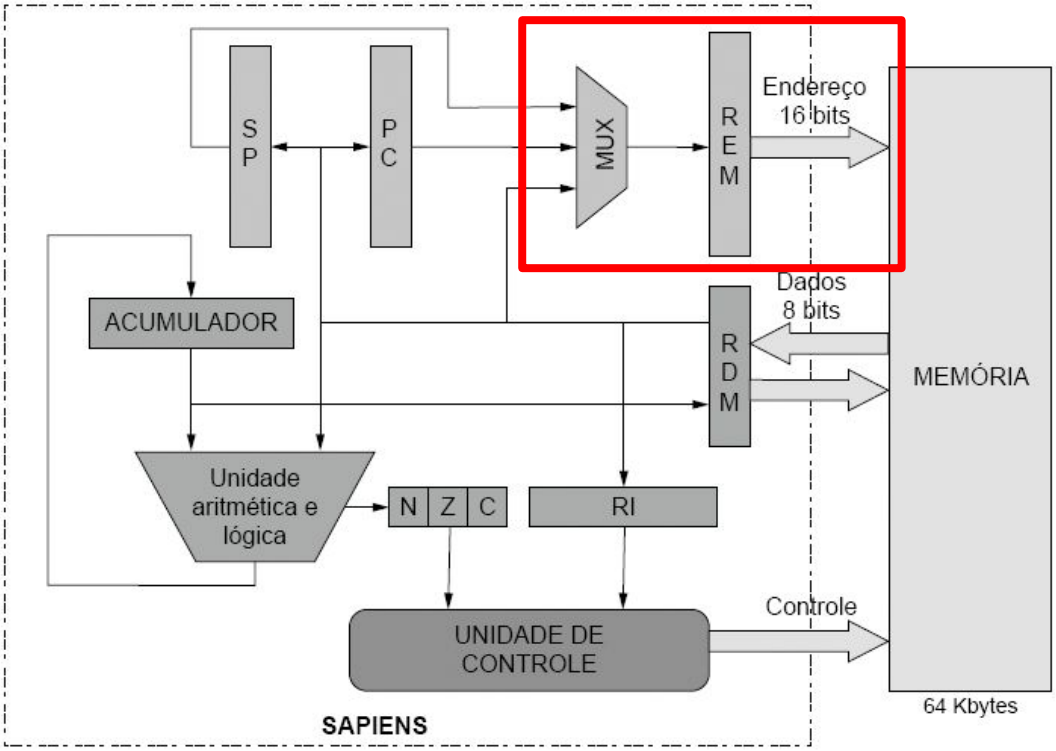
- Possui instruções de IN e OUT para realizar operações de entrada e saída em dispositivos de E/S, em um espaço de endereçamento de 256 bytes, separado do espaço de endereçamento da memória

Processador Sapiens: Detalhes da microarquitetura



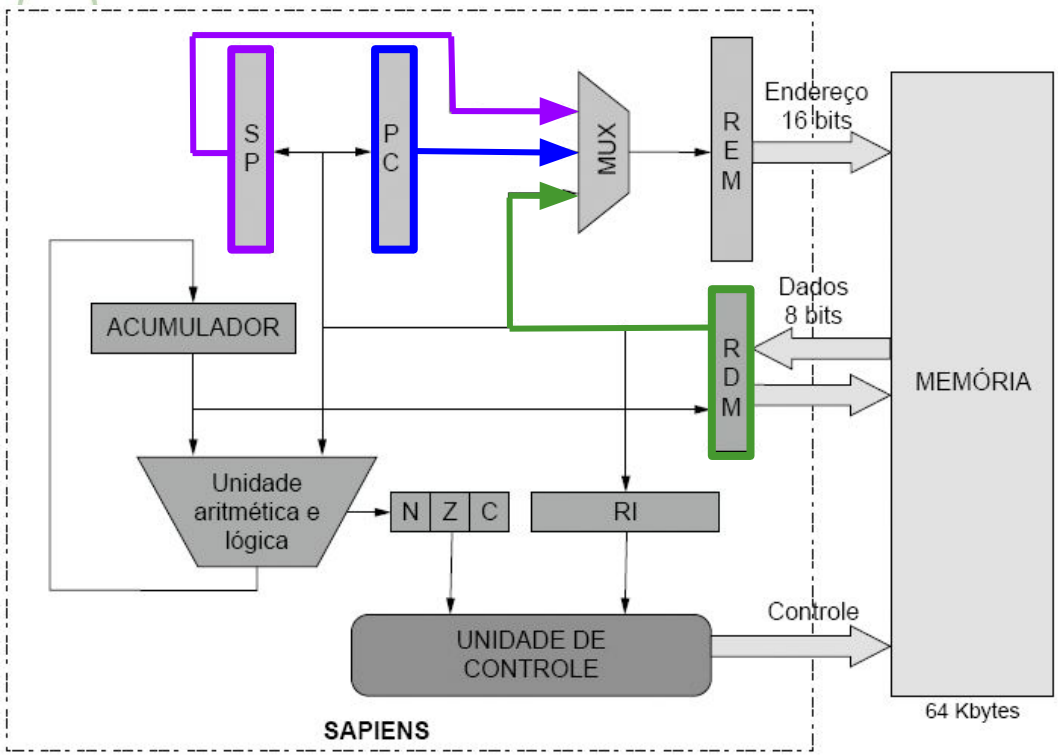
- PC (Program Counter) é o apontador da próxima instrução a ser executada pelo Processador
- O programa, que é o conjunto de instruções e dados no formato binário, é lido a partir do endereço inicial carregado (apontado pelo PC)
- O processador realiza permanentemente o ciclo de busca e execução de instruções e o valor do PC é automaticamente incrementado de 1 após cada leitura de um byte da memória, mas pode ser modificado por instruções de desvio e chamadas e retornos de procedimentos

Processador Sapiens: Detalhes da microarquitetura



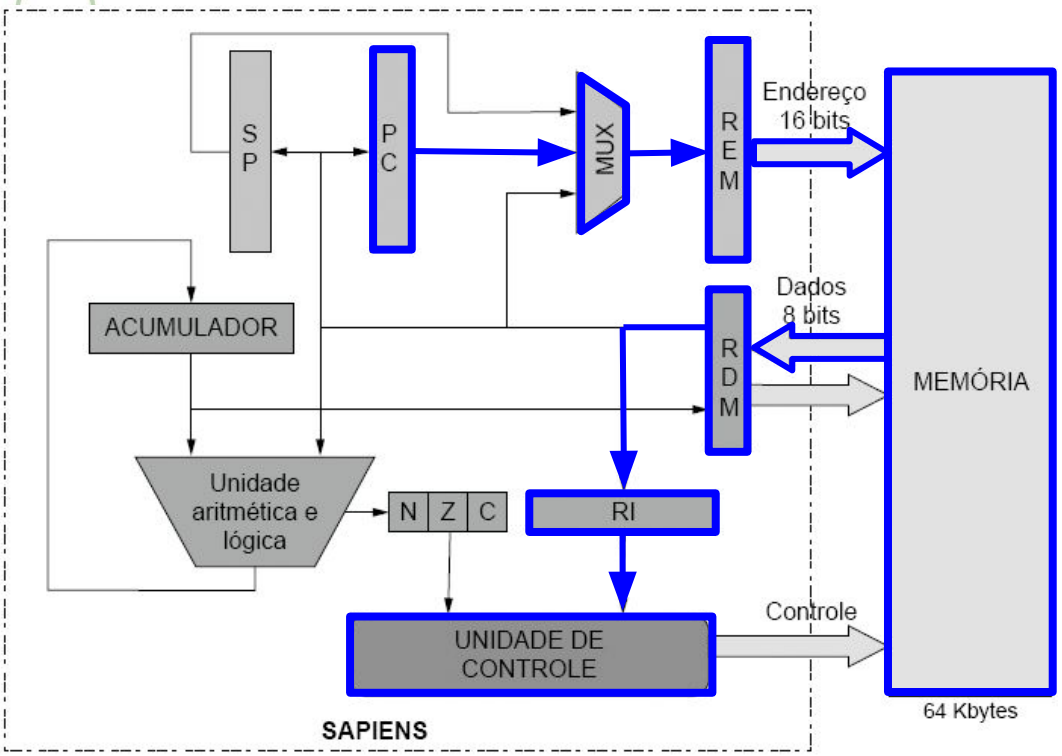
- Os acessos à memória, para a busca de dados ou instruções, são feitos a partir do endereço armazenado no registrador de endereço de memória (REM)
- O multiplexador (MUX) decide de onde virá o endereço a ser armazenado no REM

Processador Sapiens: Detalhes da microarquitetura



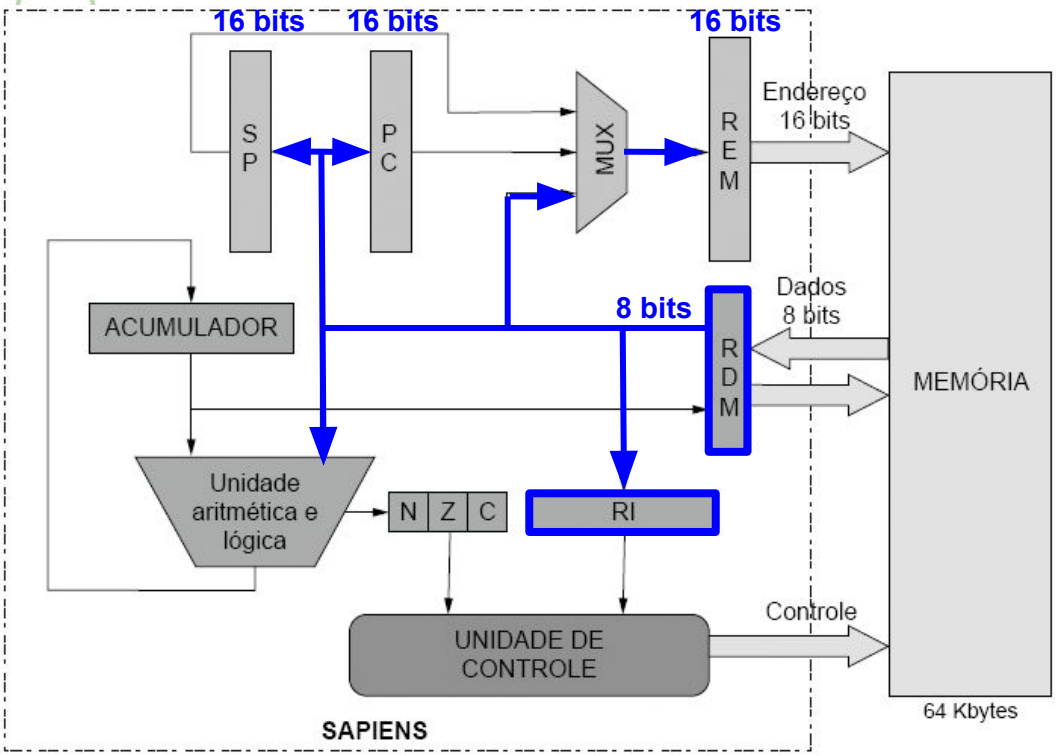
- Do PC quando o valor do REM é utilizado para acessar as instruções, dados imediatos e endereços que estão no código do programa
- Do registrador de dados da memória (RDM), quando o endereço no REM serve para acessar os operandos no modo direto ou indireto na memória (como é 16 bits, é feita em dois passos). Então, no modo direto são necessários dois acessos à memória e no modo indireto, essa operação ocorre quatro vezes
- Do apontador de pilha (SP), quando o endereço no REM serve para acessar os operandos na pilha com as instruções PUSH e POP ou para salvar e restaurar o endereço de retorno durante a chamada ou retorno de procedimento.

Processador Sapiens: Ciclo de busca de operação (OpCode)



- O ciclo de busca de instruções usa o endereço armazenado em REM (vindo do PC) para transferir as instruções da memória para o RDM e daí então para o registrador de instruções (RI)
- Ao chegar no RI, a instrução é analisada pela unidade de controle e, conforme o seu código de operação (OpCode), os caminhos de dados e registradores internos do processador são acionados adequadamente para execução da instrução.

Processador Sapiens: Ciclo de execução de instruções



- No ciclo de execução da instrução, o valor em REM serve para a busca de operandos
- Os operandos do RDM podem:
  - ser armazenado AC
  - ser o segundo operando na UAL
  - ser armazenado PC (em desvio ou de retorno de procedimento são dois acessos - 16 bits)
  - ser movido para o REM, para definir o endereço de memória do operando (16 bits). No modo indireto (16 + 16 bits)
  - ser movido para o apontador de pilha (SP), no caso da instrução LDS (16 bits).

INSTITUTO FEDERAL
Santa Catarina

Processador Sapiens: Tabela de instruções

Mnemônico	Código	Descrição
NOP	0000 0000	Não faz nada
STA ender, STA @ender	0001 000x	Armazena o acumulador (um byte) na memória
STS ender, STS @ender	0001 010x	Armazena o apontador de pilha (dois bytes) na memória
LDA #imed, LDA ender, LDA @ender	0010 00xx	Carrega o operando (um byte) no acumulador
LDS #imed16, LDS ender, LDS @ender	0010 01xx	Carrega o operando (dois bytes) no apontador de pilha (SP)
ADD #imed, ADD ender, ADD @ender	0011 00xx	Soma o acumulador com o operando (um byte)
ADC #imed, ADC ender, ADC @ender	0011 01xx	Soma o acumulador com o carry (flag C) e com o operando (um byte)
SUB #imed, SUB ender, SUB @ender	0011 10xx	Subtrai o acumulador do operando (um byte)
SBC #imed, SBC ender, SBC @ender	0011 11xx	Subtrai o acumulador do carry (flag C) e do operando (um byte)
OR #imed, OR ender, OR @ender	0100 00xx	Realiza um "ou" bit a bit entre o acumulador e o operando (um byte)
XOR #imed, XOR ender, XOR @ender	0100 01xx	Realiza um "ou exclusivo" bit a bit entre o ACC e o operando (um byte)
AND #imed, AND ender, AND @ender	0101 00xx	Realiza um "e" bit a bit entre o acumulador e o operando (um byte)
NOT	0110 0000	Complementa (inverte) os bits do acumulador.
SHL	0111 0000	Deslocamento do acumulador de um bit para a esquerda, através do carry
SHR	0111 0100	Deslocamento do ACC à esquerda, através do carry, insere 0
SRA	0111 1000	Deslocamento do ACC à esquerda, através do carry, duplica o MSB

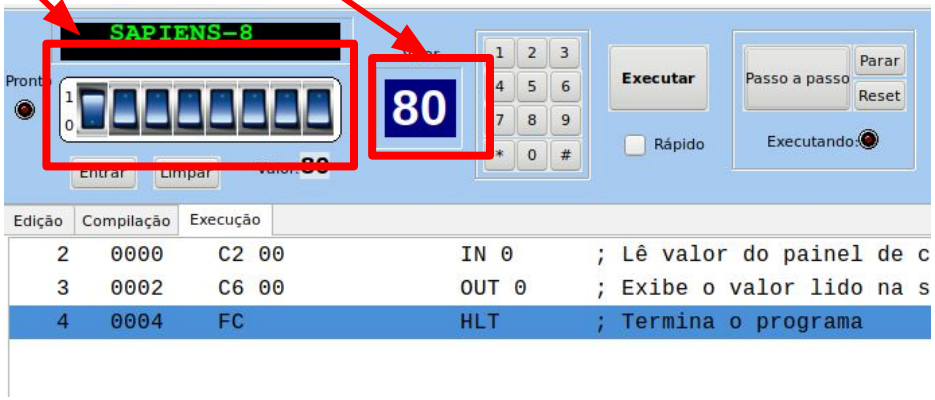
INSTITUTO FEDERAL
Santa Catarina

Processador Sapiens: Tabela de instruções

Mnemônico	Código	Descrição
JMP ender, JMP @ender	1000 000x	Desvia a execução do programa para o endereço
JN ender, JN @ender	1001 000x	Desvia a execução do programa para o endereço, apenas se N = 1
JP ender, JP @ender	1001 010x	Desvia a execução do programa para o endereço, apenas se N = 0 e Z = 0
JZ ender, JZ @ender	1010 000x	Desvia a execução do programa para o endereço, apenas se Z = 1
JNZ ender, JNZ @ender	1010 010x	Desvia a execução do programa para o endereço, apenas se Z = 0
JC ender, JC @ender	1011 000x	Desvia a execução do programa para o endereço, apenas se C =1
JNC ender, JNC @ender	1011 010x	Desvia a execução do programa para o endereço, apenas se C = 0
IN ender8	1100 0000	Carrega no acumulador o valor lido no endereço de E/S
OUT ender8	1100 0100	Descarrega o conteúdo do acumulador no endereço de E/S
JSR ender, JSR @ender	1101 000x	Desvia para procedimento
RET	1101 1000	Retorno de procedimento
PUSH	1110 0000	Coloca o conteúdo do acumulador no topo da pilha
POP	1110 0100	Retira o valor que está no topo da pilha e coloca no acumulador
TRAP ender, TRAP @ender	1111 0000	Instrução para emulação de rotinas de E/S pelo simulador
HLT	1111 1111	Para a máquina

Processador Sapiens: Exemplo de assembly: interação com I/O

```
ORG 0
    IN 0        ; Lê valor do painel de chaves
    OUT 0       ; Exibe o valor lido na saída
    HLT         ; Termina o programa
END 0
```

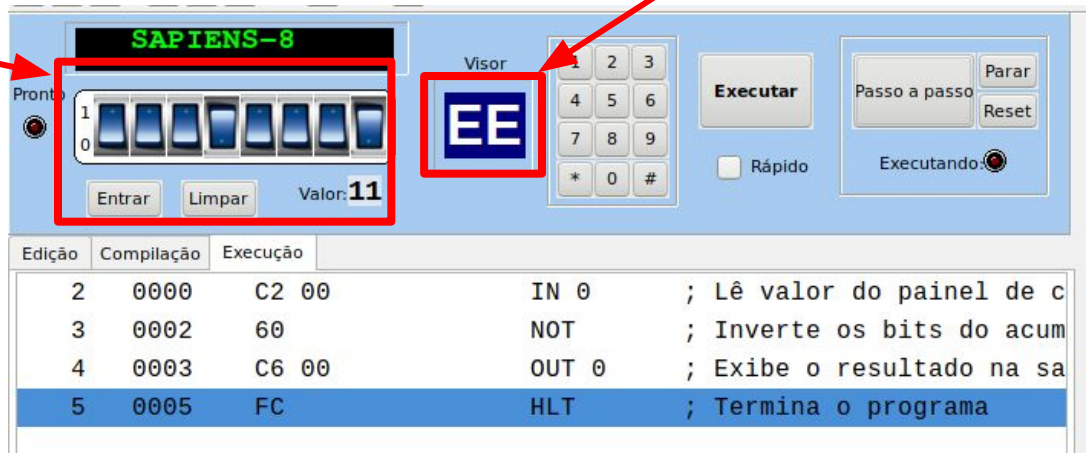


Processador Sapiens: Exemplo de assembly: interação com I/O

```
ORG 0
    IN 0        ; Lê valor do painel de chaves
    NOT         ; Inverte os bits do acumulador
    OUT 0       ; Exibe o resultado na saída
    HLT         ; Termina o programa
END 0
```

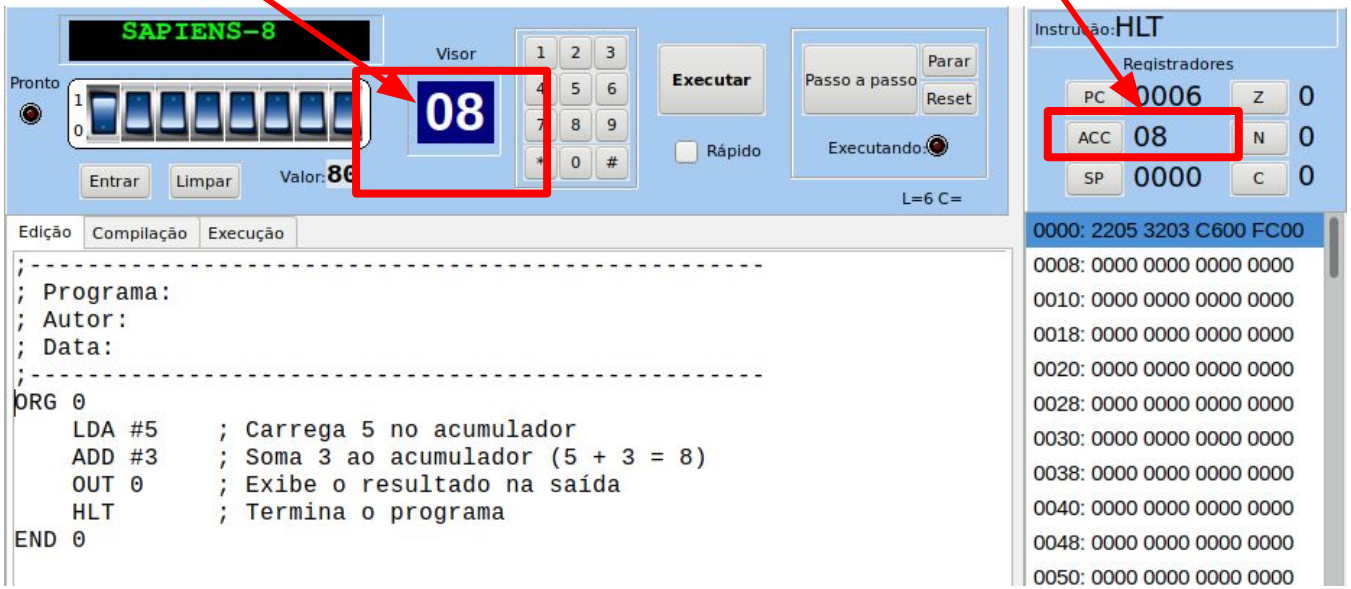
O inverso de
0x11 (0b 0001 0001
é 0xEE (0b 1110 1110

Nesse exemplo a entrada tem 0x11



Processador Sapiens: Exemplo de assembly: soma imediata

```
ORG 0
    LDA #5      ; Carrega 5 no acumulador
    ADD #3      ; Soma 3 ao acumulador (5 + 3 = 8)
    OUT 0       ; Exibe o resultado na saída
    HLT         ; Termina o programa
END 0
```

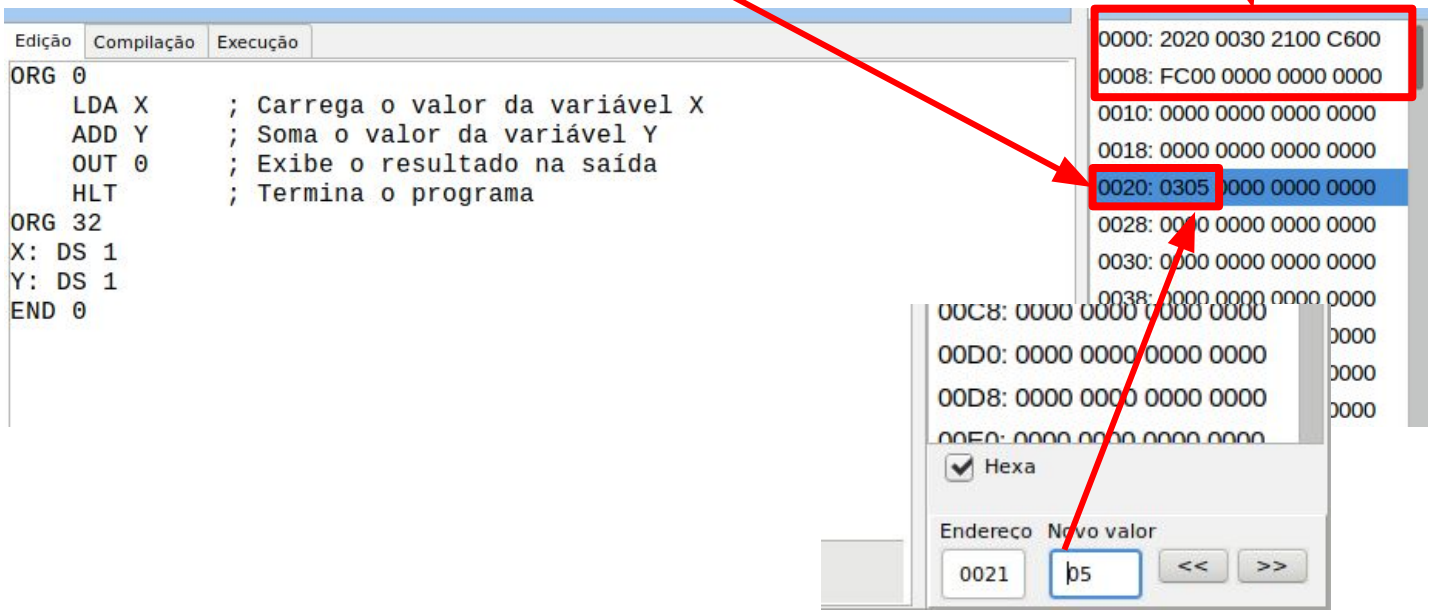


Processador Sapiens: Exemplo de assembly: soma variáveis

```
ORG 0
    LDA X      ; Carrega o valor da variável X
    ADD Y      ; Soma o valor da variável Y
    OUT 0      ; Exibe o resultado na saída
    HLT        ; Termina o programa
ORG 32
X: DS 1
Y: DS 1
END 0
```

Programa

Variáveis a partir do endereço 32 (0x20)



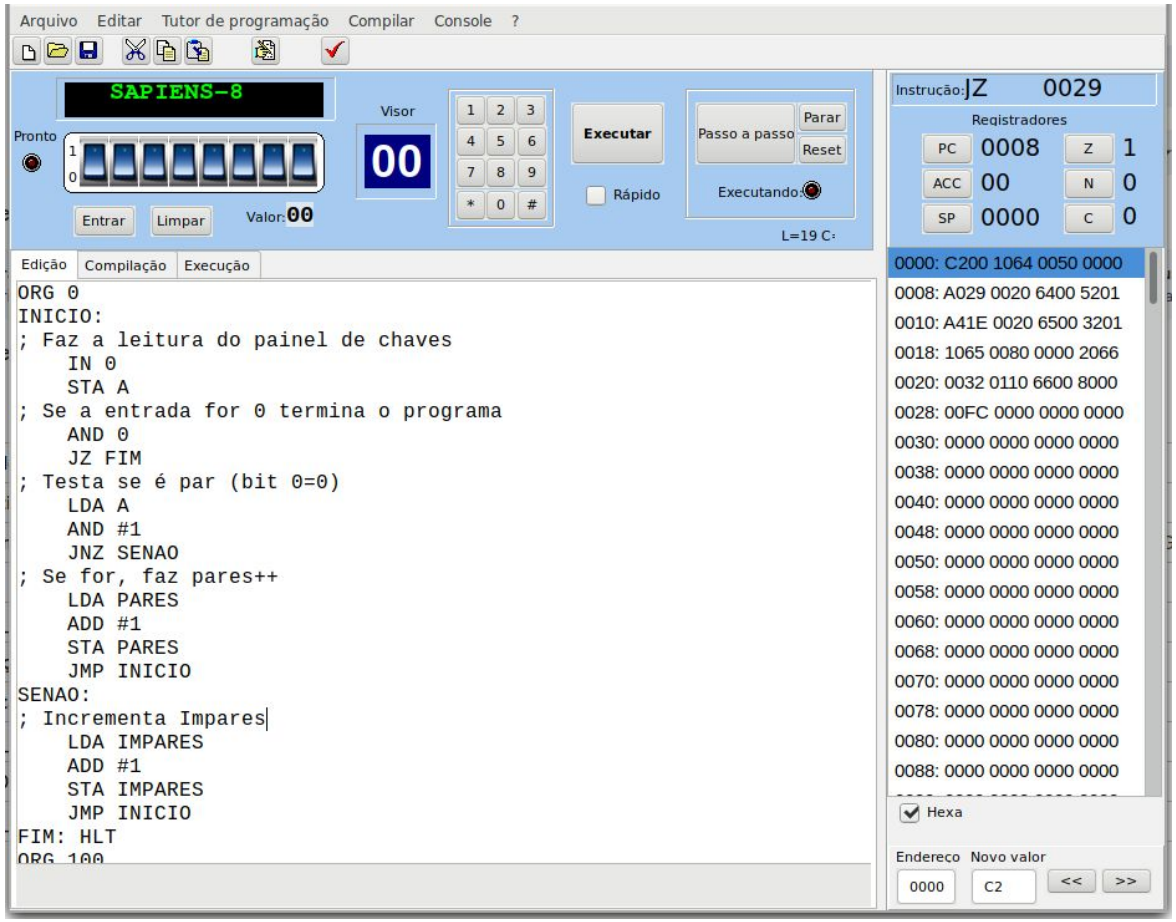
Processador Sapiens: Exemplo de assembly: multiplica com somas

```
ORG 0
    LDA #0     ; Inicializa acumulador com 0
    STA RES    ; Armazena resultado inicial
    LDA #4     ; Contador para 4 iterações
    STA CONT   ; Armazena contador
LOOP:
    LDA CONT   ; Carrega contador
    JZ FIM     ; Se contador = 0, termina
    LDA RES    ; Carrega resultado atual
    ADD #3     ; Soma 3 ao resultado
    STA RES    ; Armazena novo resultado
    LDA CONT   ; Carrega contador
    SUB #1     ; Decrementa contador
    STA CONT   ; Armazena contador
    JMP LOOP   ; Volta ao laço
FIM:
    LDA RES    ; Carrega resultado final
    OUT 0      ; Exibe resultado
    HLT        ; Termina
ORG 100
RES: DS 1      ; Variável para resultado
CONT: DS 1     ; Variável para contador
END 0
```

Operação 3 x 4 = 12

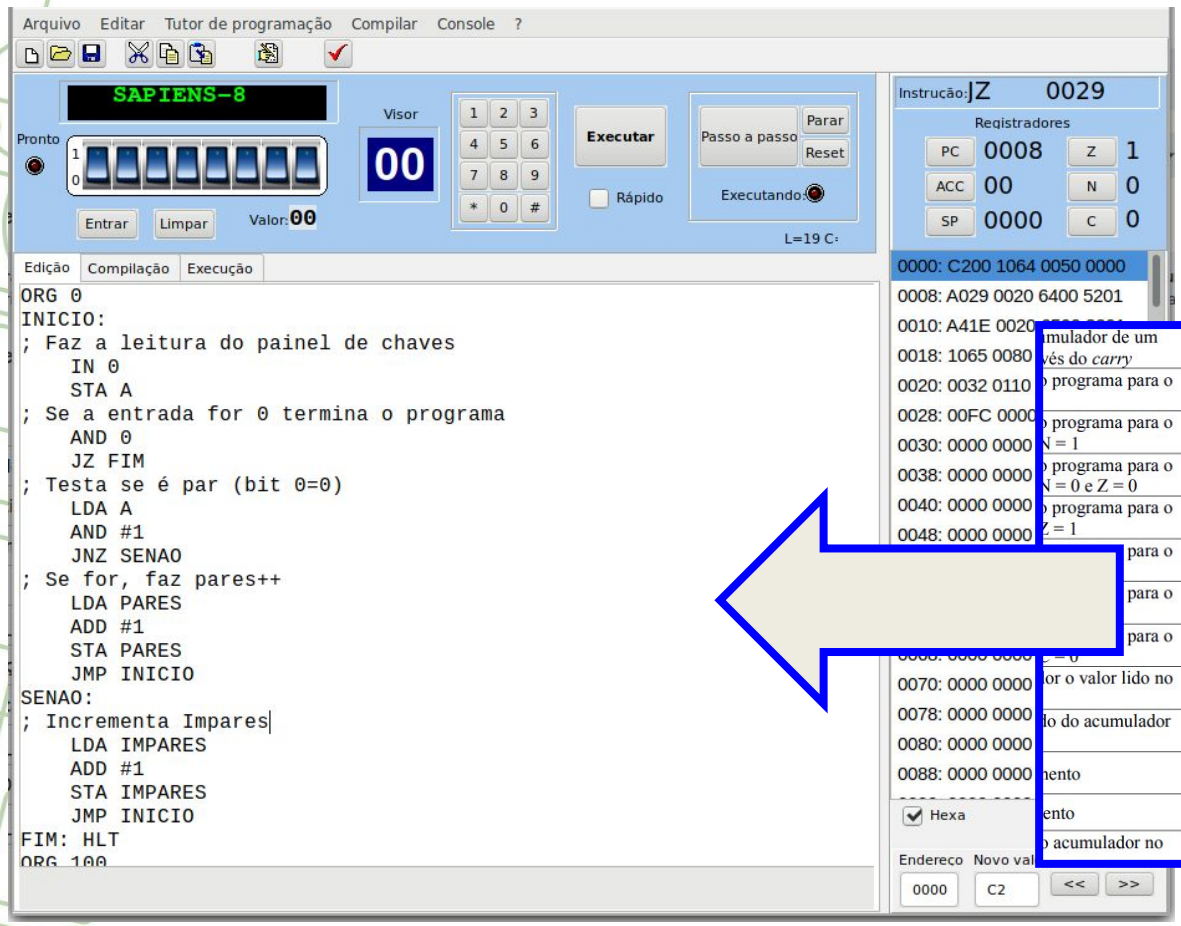
Adiciona 3 ao acumulador por 4 vezes

Processador Sapiens: Simulador



Simulador e manual disponíveis em:
https://sourceforge.net/projects/simus/files/

Processador Sapiens: Rodando um exemplo



Carregando o programa exemplo que vem no manual do SimuS. Este algoritmo realiza a contagem de entradas PARES e ÍMPARES.

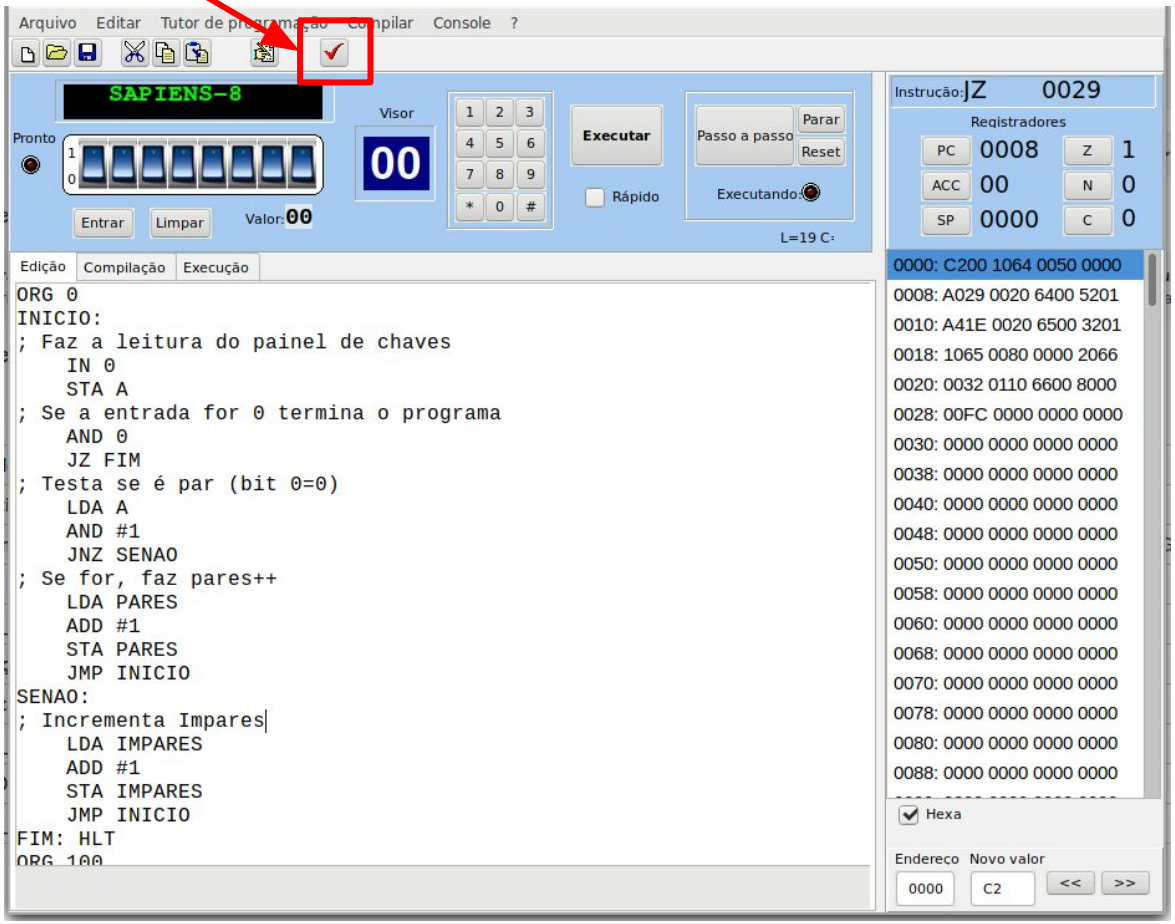
Na maioria são mnemônicos e comandos com sin
simplificada e de fácil utilização. A seguir um exempl
programa em linguagem de montagem para o process
Sapiens:

```
ORG 0
INICIO:
; Faz a leitura do painel de chaves
        IN  0
        STA A
; Se a entrada for 0 termina o programa
        AND 0
        JZ  FIM
; Testa se é par (bit 0=0)
        LDA A
        AND #1
        JNZ SENAO
; Se for, faz pares++
        LDA PARES
        ADD #1
        STA PARES
        JMP INICIO
SENAO:
```

Simulador e manual disponíveis em:
https://sourceforge.net/projects/simus/files/

Processador Sapiens: Rodando um exemplo

Compile
o código

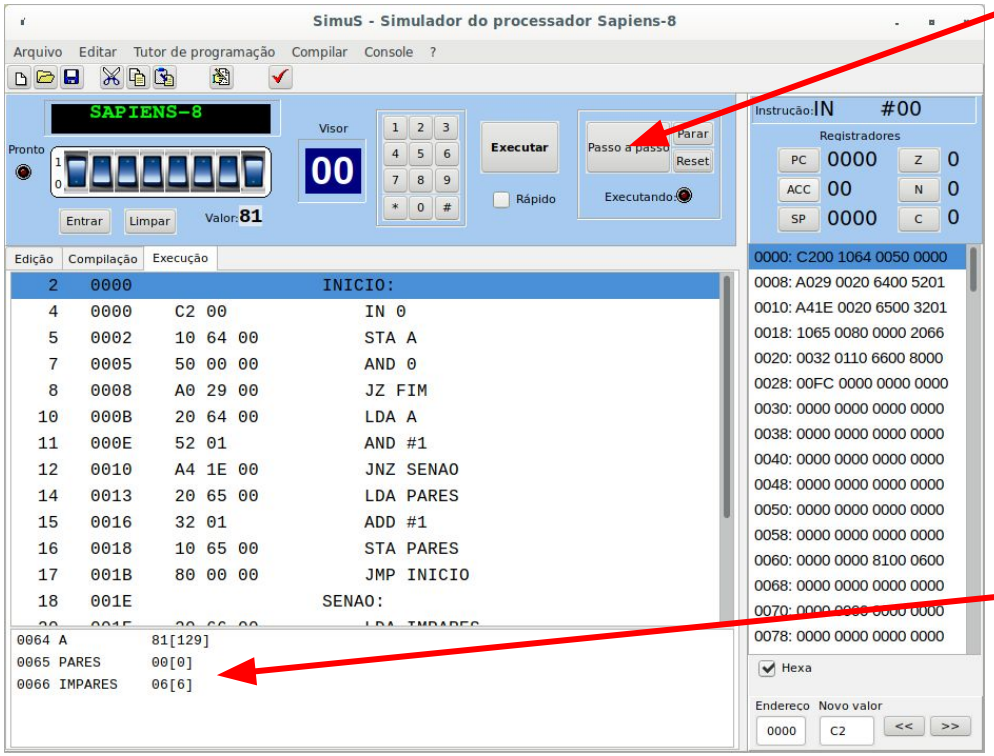


Processador Sapiens: Rodando um exemplo

Altere as chaves de entrada para que o valor seja 0x81

O programa encerra se o valor de entrada for 00

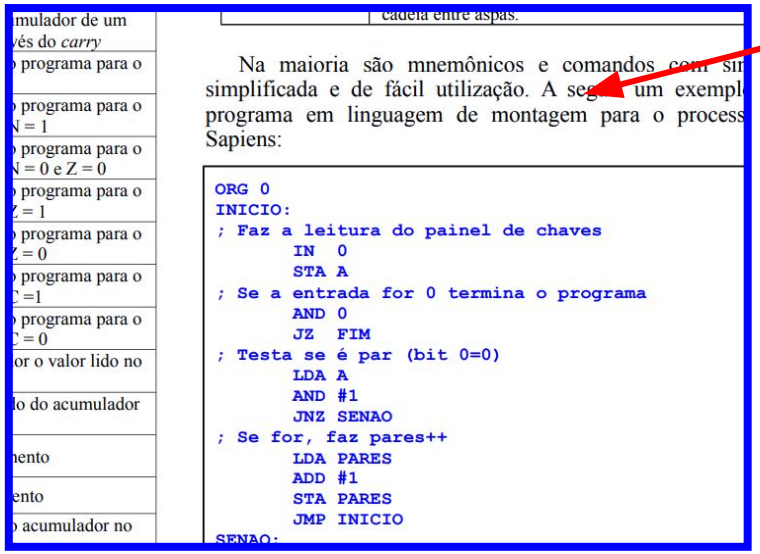
Clique em passo a passo para avançar a execução



Observe a contagem aumentando

Processador Sapiens: Rodando um exemplo

Varie a entrada para outros valores, por exemplo, 0x80 e veja o contador de números PARES incrementar.



...mulador de um
...vés do *carry*
...programa para o
...programa para o
...N = 1
...programa para o
...N = 0 e Z = 0
...programa para o
...Z = 1
...programa para o
...Z = 0
...programa para o
...Z =1
...programa para o
...Z = 0
...or o valor lido no
...do acumulador
...mento
...ento
...acumulador no

...cadeia entre aspas.

Na maioria são mnemônicos e comandos com sin... simplificada e de fácil utilização. A seg... um exempl... programa em linguagem de montagem para o process... Sapiens:

```
ORG 0
INICIO:
; Faz a leitura do painel de chaves
        IN   0
        STA A
; Se a entrada for 0 termina o programa
        AND 0
        JZ   FIM
; Testa se é par (bit 0=0)
        LDA A
        AND #1
        JNZ SENAO
; Se for, faz pares++
        LDA PARES
        ADD #1
        STA PARES
        JMP INICIO
SENAO:
```

Leia o código atentamente, utilize o livro “Arquitetura e Organização de Computadores – Uma Introdução” para interpretar cada instrução.