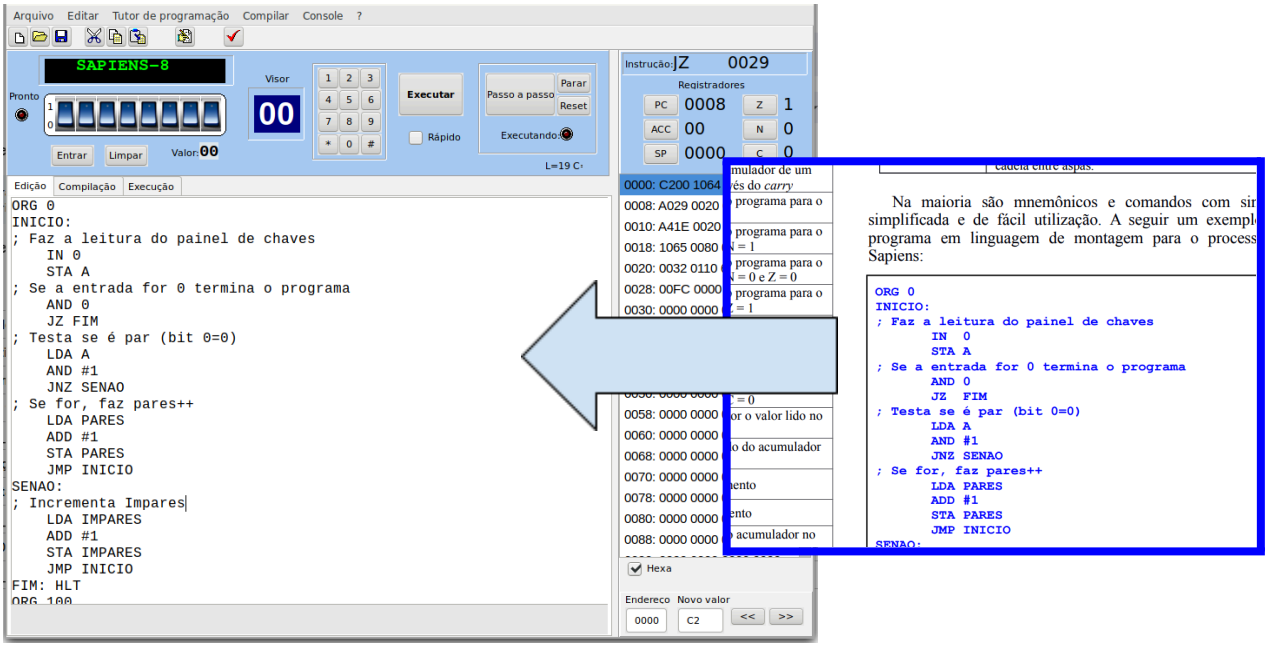
LABORATÓRIO - Processadores - SimuS Sapiens

1. Faça o download e rode o simulador Simus do Processador Sapiens
   a. Baixe o manual do Simus e abra o livro digital "Arquitetura e Organização de Computadores – Uma Introdução" na página 134 onde estão descritas as instruções do Sapiens (ambos disponíveis em nossa turma virtual do SIGAA)
   b. Considerando que você está utilizando uma máquina Linux 64 bits, faça o download do arquivo simus_lin64 em: https://sourceforge.net/projects/simus/
   c. Após baixar o arquivo, é necessário marcá-lo como executável, em um terminal, dentro da pasta onde o arquivo foi baixado (ex.: /home/aluno/Downloads) digite: chmod +x simus_lin64
   d. Execute o programa com o comando: ./simus_lin64
   e. Você deverá ver a tela a seguir, porém, com o campo texto da aba edição em branco. Abra então o manual do SimuS e copie e cole o programa de exemplo.



   f. Após copiar o programa, clique na opção de menu "compilar", isso deve gerar resultados na aba de "Compilação".
   g. Clique agora em "Passo a passo" para observar o programa sendo executado.
   h. Você pode reiniciar o programa a qualquer momento, clicando em reset (faz ele voltar para o início do programa).
   i. Esse programa exemplo conta numerais de entrada pares e ímpares, porém, caso o numeral de entrada seja zero, ele finaliza. Para ver a contagem funcionar, altere as chaves para que a entrada seja, por exemplo, 0x81 e rode novamente o programa:

j. Você poderá ver a variável “IMPARES” exibida na parte inferior da janela sendo incrementada a cada vez que o programa atesta que a entrada está com o numeral 0x81. Mude a entrada para outros valores, pares e impares e veja as variáveis “PARES” e “IMPARES” modificares conforme a entrada.

2. Explore o programa exemplo e reflita sobre as proposições a seguir:
   a. Observe que a instrução “ORG 0” faz com que o código do editor seja gravado a partir do primeiro endereço de memória (0x0000). Vamos alterar isso para ser gravado a partir de 0x0030, sobrescrevendo por “ORG 48”. Recompile e rode. O que ocorreu? Qual a relação entre os numerais 48 e 0x0030?
   b. Volte para “ORG 0” e recompile. Perceba que a instrução “IN 0” copia para o ACC o valor das chaves de entrada. Coloque nas chaves o valor 81 (0x81), execute o passo desta instrução e observe no canto direito superior que ACC armazena o valor 81.
   c. A instrução “STA A” serve para fazer um “backup” das entradas. Observe que essa instrução modifica a variável “A” exibida na barra inferior esquerda da aba “Execução”. As entradas ficam, portanto, salvas na variável “A”. Por que fazer esse “backup” é relevante? Por que, na linha mais adiante onde temos “LDA A” não se faz a leitura das entradas (IN 0) novamente? Em qual endereço de memória a variável “A” está sendo gravada?
   d. Agora observe o código “AND 0”. Essa linha é curiosa, você não acha? Lendo o livro e manual, o que está fazendo exatamente? Você não achou estranho ter sido feito “AND 0” e não “AND 255” ou “OR 0”?
   e. O que a instrução “JZ FIM” faz? Quais são os possíveis condicionantes e desdobramentos?
   f. O que a instrução “AND #1” está fazendo neste código? Se usássemos “AND #255” teríamos o mesmo resultado?
   g. Se ao invés de termos “AND #1” seguido de “JNZ SENAO”, tivéssemos “SHR” seguido de “JC SENAO”, o algoritmo se comportaria diferentemente? Estas abordagens mudam alguma coisa em termos de tamanho do código do programa?
   h. Em quais endereços de memória as variáveis PARES e IMPARES estão armazenadas? É possível observar a alteração destes endereços conforme as entradas que introduzimos?

3. Explore e reflita sobre o código a seguir:

```
ORG 0
INICIO:
   LDA #10
   STA CONT
LOOP10:
   LDA CONT
   SUB #1
   STA CONT
   JNZ LOOP10
FIM:
   HLT
ORG 32
CONT: DS 1
END 0
```

   a. O que esse programa faz?
   b. Qual o tamanho total deste programa em Bytes?
   c. O que está sendo gravado no endereço 0x0020?
   d. Se você alterar a instrução “LDA #10” para “LDA #15” o que ocorre?

4. Um autor modificou esse programa conforme código a seguir. O que mudou?

```
ORG 0
INICIO:
   LDS #65
   LDA #10
   PUSH
LOOP10:
   POP
   SUB #1
   PUSH
   JNZ LOOP10
FIM:
   HLT
END 0
```

5. Baseado no código anterior, um autor modificou o programa conforme código a seguir. O que mudou? Qual possível utilidade disso?

```
ORG 0
INICIO:
   LDS #65
   LDA #10
   PUSH
LOOP10:
   LDA #255
   OUT #0
   NOP
   NOP
   NOP
   LDA #0
   OUT #0
   POP
   SUB #1
   PUSH
   JNZ LOOP10
FIM:
   HLT
END 0
```

6. VALE BIT: Sabendo que podemos fazer a multiplicação de A x B, somando A por B vezes, faça um programa que realiza a multiplicação de duas variáveis (“A” e “B”) e escreve o resultado no endereço de uma terceira variável chamada “R”.