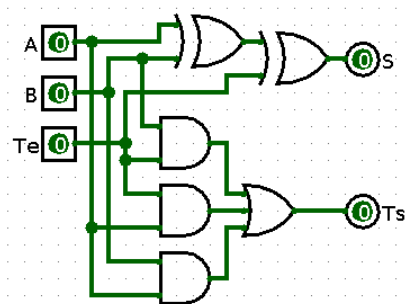
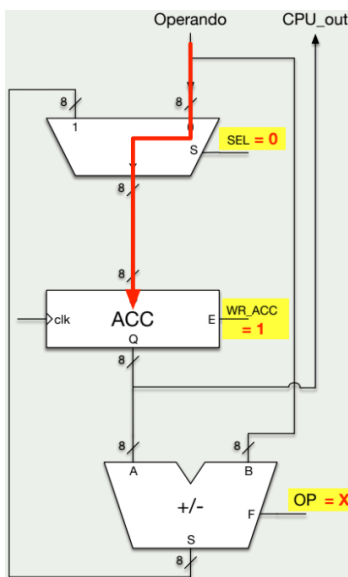


## Processadores - Exercícios de Fixação

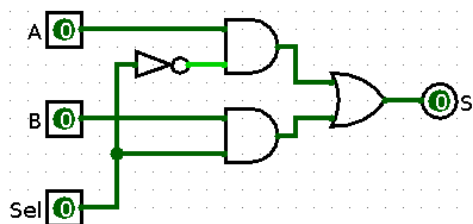
1. Explique o que diferencia um computador de uma calculadora.
2. Descreva uma contribuição de Alan Turing para a computação.
3. Descreva uma contribuição de Von Neumann para a computação.
4. Quais os principais passos que um processador executa em cada ciclo?
5. Explique o que é o Acumulador, para que serve e por que é importante.
6. Para que serve o circuito a seguir? Monte sua tabela verdade.



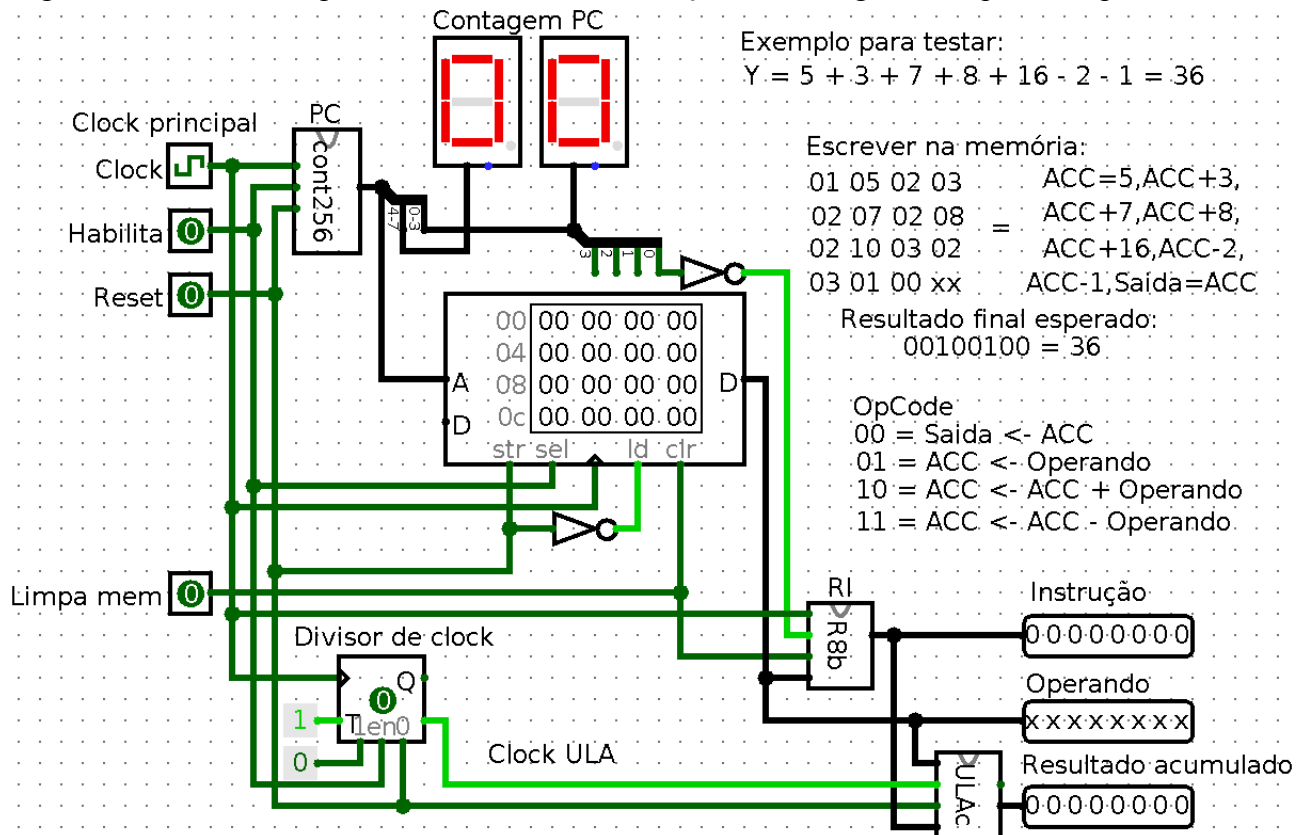
7. Considerando o diagrama a seguir, descreva a operação que está ocorrendo passo a passo, explicando também qual seu propósito e resultado esperado.



8. Considere o circuito a seguir. O que ele faz? Como se chama? Descreva seu funcionamento nos possíveis estados.



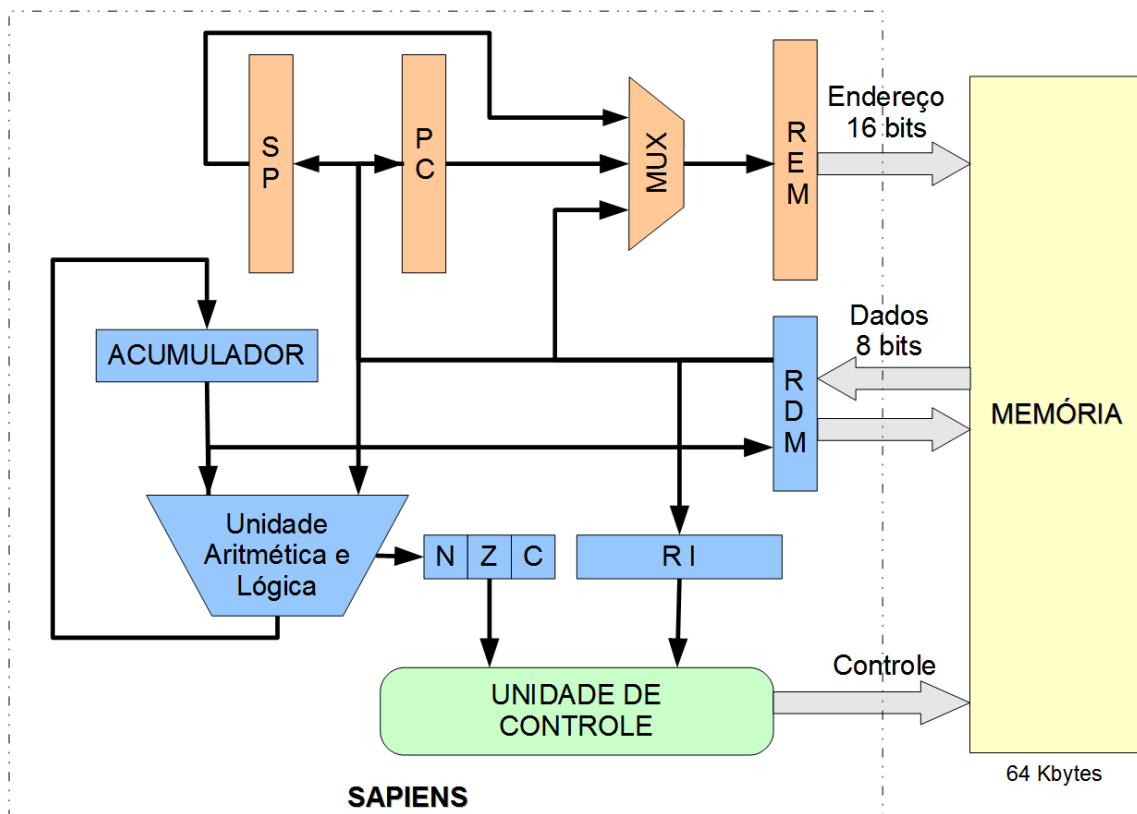
9. Considere o diagrama a seguir de uma microarquitetura. Considere que o componente PC é um contador de módulo 256, que RI é um registrador de 8 bits e ULAc é uma Unidade Lógica e Aritmética capaz de realizar as 4 instruções da imagem no quadro OpCode.



Resposta:

- Explique a função dos componentes PC e RI desta microarquitetura.
- O que ocorre se habilita = 1 e fizermos múltiplos ciclos de clock quando a memória estiver com tudo 00 (conforme a figura ilustra)?
- Agora considere que a memória foi escrita com os dados do quadro “Escrever na memória”. Considere que foi dado o Reset. Qual posição PC estará apontando? Qual o Valor em RI e no Resultado acumulado?
- Agora considere que foi feito um ciclo de clock. Qual posição PC estará apontando? Qual o Valor em RI e no Resultado acumulado?
- Agora considere que foram feitos mais três ciclos de clock (somando os ciclos, estamos no quarto ciclo). Qual posição, PC estará apontando? Qual o Valor em RI e no Resultado acumulado?
- Considerando que a entrada de dados da ULA está conectada ao barramento de dados da memória e são lidas instruções e operandos, explique como este projeto consegue diferenciar o que é operando e o que é instrução.
- Esta microarquitetura está escrevendo dados na memória principal? Justifique.

10. Considere a microarquitetura do processador didático Sapiens, conforme diagrama a seguir:



- Qual a diferença entre SP e PC? Explique cada um deles.
  - Qual a função de REM e RDM?
  - O que significam os componentes N, Z e C? Na prática, para que servem?
  - Considerando os modos de endereçamento, é correto afirmar que no modo indireto deve-se informar o endereço de memória que contem um operando? Justifique e compare com o modo direto.
11. Descreva o funcionamento do algoritmo a seguir:

ORG 0

INICIO:

LDA #10

STA CONT

LOOP10:

LDA CONT

SUB #1

STA CONT

JNZ LOOP10

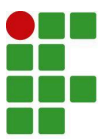
FIM:

HLT

ORG 32

CONT: DS 1

END 0

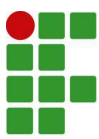


12. Elabore um algoritmo em Assembly da microarquitetura Sapiens que executa a operação  $7 - 4$  (sete menos quatro).
13. Elabore um algoritmo em Assembly da microarquitetura Sapiens que lê o valor do painel, calcula o dobro deste valor e apresentando o resultado da soma na saída.
14. Elabore um algoritmo em Assembly da microarquitetura Sapiens que lê o valor do painel de chaves e realiza a operação lógica E (AND) entre o valor lido e a constante 0x03, exibindo na saída o resultado desta operação.
15. Elabore um algoritmo em Assembly da microarquitetura Sapiens que exibe na saída os valores de 0 a 3 um a um parando o programa após exibir o valor 3.



## Gabarito

1. Um computador é um dispositivo programável que armazena e processa dados e instruções na memória, permitindo a execução de tarefas variadas e complexas. Uma calculadora, por outro lado, é um dispositivo especializado para realizar cálculos aritméticos predefinidos, com funcionalidade limitada e sem capacidade de reprogramação geral.
2. No artigo intitulado “On Computable Numbers, with an Application to the Entscheidungsproblem” de 1936, Alan Turing desenvolveu a ideia do que hoje é chamado de Máquina de Turing, um modelo teórico que formaliza o que significa computar. A Máquina de Turing é um dispositivo abstrato com uma fita infinita, uma cabeça de leitura/escrita e um conjunto de regras que definem como processar símbolos com base em estados. Esse modelo demonstrou que qualquer problema computável pode ser resolvido por uma máquina com um conjunto finito de instruções. A Máquina de Turing influenciou diretamente o projeto de computadores modernos e o desenvolvimento da ciência da computação como disciplina.
3. No relatório intitulado "First Draft of a Report on the EDVAC" de 1945, John von Neumann propôs o que hoje é chamado de arquitetura de Von Neumann, um modelo que define a estrutura básica dos computadores modernos. Essa arquitetura introduziu o conceito de um programa armazenado na memória, onde instruções e dados são guardados na mesma memória, acessados sequencialmente por uma unidade de controle e processados por uma unidade lógico-aritmética.
4. Grosseiramente em cada ciclo um processador obtém a instrução a ser executada, atualiza o PC, decodifica a instrução (OpCode), busca operando(s), executa a instrução e armazena resultados.
5. O acumulador é um registrador que é parte central de uma ULA, recebendo normalmente resultados parciais de operações. O acumulador é utilizado em praticamente todas as operações executadas em um processador. O acumulador serve, portanto, para receber dados que serão operados ou para guardar resultados parciais de operações.
6. É um somador completo, serve para somar 1 bit com outro, sendo um bloco utilizado para montagem de somadores de palavras de vários bits, por exemplo, de 8 bits, onde são utilizados 7 destes. Ver tabela no material didático fornecido.
7. Nessa operação o Mux está ativado para que em sua saída seja copiado o valor da entrada de operando, levando este dado ao acumulador que está configurado para escrita (escrevendo em ACC o operando disponível). O propósito, portanto, é escrever em ACC o valor em operando.



8. Um mux, se  $sel = 0$  a saída terá o valor da entrada A, se  $sel = 1$  a saída terá o valor da entrada B.

9. a) PC ou Program Counter é um contador de módulo 256 que armazena o endereço da próxima instrução a ser buscada na memória para execução. Ele determina a sequência do programa, incrementando automaticamente após cada ciclo de busca para apontar para a próxima instrução. RI ou Registrador de Instrução um registrador de 8 bits que armazena a instrução atual buscada da memória durante o ciclo de busca. Como a instrução desta arquitetura contém apenas o OpCode (não contém operando), o RI armazena apenas o OpCode.

9. b) A instrução 00 que seria lida nesta situação hipotética prevê que a saída ficará mostrando o conteúdo do ACC, que não vai ser alterado já que esta instrução não modifica o ACC, portanto, a única modificação que será percebida será o próprio incremento do PC.

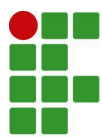
9. c) Após o reset o PC vai apontar para a posição 0x00 (zero). Como nesta arquitetura o primeiro byte (bytes pares) é uma instrução, o valor de RI será 0x01 (conforme código informado) e o resultado acumulado num primeiro momento é zero (considerando que o ACC iniciou após o reset com zero).

9. d) Após um ciclo PC apontará para 0x01. O valor de RI permanecerá 0x01 e o resultado acumulado também se manterá em zero (considerando que após o reset ACC era zero). De fato, o valor acumulado só será modificado após um segundo ciclo de clock. O que mudou nesta nova condição em relação a anterior é que a saída operando estará mostrando 0x05.

9. e) Após completar 4 ciclos de clock PC apontará 0x04 (cada ciclo fez pular e endereço percorrendo 0x00, 0x01, 0x02, 0x03, chegando a 0x04). RI terá o valor 0x02 da operação soma. O resultado acumula a carga do valor 5 e a soma com 3 que resulta em 8 (sendo este o valor acumulado)

9. f) Esta microarquitetura diferencia instruções e operandos pela posição de cada um na memória, sendo que as instruções estão sempre em posições de memória de número PAR (ex.: 0x00, 0x02, 0x04, ...) e os operandos sempre em posições de memória IMPAR (ex.: 0x01, 0x03, 0x05, ...). Desta forma, os dados em endereços pares vão para o RI e os dados em endereços ímpares são carregados na ULA como operandos. O salto entre endereços pares e ímpares é feito da ativação de RI quando o endereço termina com 0 (zero = PAR) e a ULA faz esse salto através de um divisor de frequência (Flip-Flop tipo T utilizado).

9. g) Esta microarquitetura não faz escrita na memória, apenas leitura. Isso pode ser observado pela configuração fixa da memória como leitura (o pino STR está fixo em zero, sendo que para se escrever na memória esse pino precisaria ficar em 1) e pelo fato da entrada de dados da memória não estar ligada a nenhuma fonte de dados, além disso, essa



arquitetura não possui nenhuma instrução que tenha sido projetada para escrever em memória (essa arquitetura apenas consegue escrever no registrador ACC).

10. a) PC ou Program Counter é o contador de programa que armazena o endereço da próxima instrução a ser buscada na memória para execução. Ele guia a sequência do programa., já o SP ou Stack Pointer é o ponteiro de pilha que aponta para o topo da pilha na memória, usado para gerenciar chamadas de sub-rotinas e armazenamento temporário de dados.

10. b) REM ou Registrador de Endereço de Memória é um registrador que armazena o endereço da memória a ser acessado (leitura ou escrita), já o RDM ou Registrador de Dados de Memória é um registrador que armazena os dados lidos da memória ou a serem escritos na memória.

10. c) N ou Negativo é um sinal que indica se o resultado da última operação é negativo (bit de sinal = 1). Z ou Zero é um sinal que indica se o resultado da última operação é zero. C ou Carry é um sinal que indica se houve transporte (carry) ou empréstimo na última operação aritmética. Na prática, esses flags são usados para controle de fluxo (ex.: desvios condicionais como JZ ou JNZ) e para verificar resultados de operações (ex.: overflow, zero ou sinal).

10. d) Sim, é correto afirmar que no modo indireto o operando informado é um endereço de memória que contém o endereço do dado a ser usado, um exemplo é a instrução LDA @100 que acessa o endereço armazenado na posição 100, e então usa o valor nesse endereço, ou seja, o valor de fato está em outro local (aquele indicado no endereço 100). Já no modo direto o operando é o próprio endereço de memória onde o dado está armazenado, por exemplo, LDA 100 carrega diretamente o valor na posição 100, ou seja, o modo indireto adiciona mais um salto, enquanto o direto acessa o dado diretamente no endereço especificado. O indireto é útil para tabelas ou ponteiros, mas é mais lento devido ao acesso extra à memória.



11.

```
ORG 0          ; Inicia o programa no endereço 0
INICIO:        ; Rótulo de início do programa
    LDA #10     ; Carrega o valor imediato 10 no acumulador
    STA CONT    ; Armazena o valor do ACC em CONT (end. 32)
LOOP10:        ; Rótulo de início do laço
    LDA CONT    ; Carrega o valor de CONT no ACC
    SUB #1      ; Subtrai 1 do valor do ACC
    STA CONT    ; Armazena o novo valor em CONT
    JNZ LOOP10  ; Se ACC != zero, volta para LOOP10
FIM:           ; Rótulo que marca o fim do programa
    HLT        ; Encerra a execução do programa
ORG 32         ; Define o endereço 32 para as variáveis
CONT: DS 1     ; Reserva 1 byte para a variável CONT
END 0          ; Marca do algoritmo
```

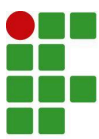
Resumo em uma linha: O algoritmo inicializa um contador com 10 e o decrementa até 0, parando a execução quando o contador chega a 0.

12.

```
ORG 0
    LDA #7      ; Carrega 7 no acumulador
    SUB #4      ; Subtrai 4 do acumulador (7 - 4 = 3)
    OUT 0       ; Exibe o resultado na saída
    HLT        ; Termina o programa
END 0
```

13.

```
ORG 0
    IN 0        ; Lê valor do painel de chaves
    STA TEMP    ; Armazena valor em TEMP
    ADD TEMP    ; Soma o valor a si mesmo (dobro)
    OUT 0       ; Exibe o resultado
    HLT        ; Termina o programa
ORG 100
TEMP: DS 1      ; Variável temporária
END 0
```



14.

```
ORG 0
    IN 0      ; Lê valor do painel de chaves
    AND #3    ; Aplica AND com a constante 3
    OUT 0     ; Exibe o resultado
    HLT       ; Termina o programa
END 0
```

15.

```
ORG 0
    LDA #0    ; Inicializa acumulador com 0
    STA TEMP  ; Armazena temporariamente
LOOP:
    LDA TEMP  ; Carrega valor
    OUT 0     ; Exibe o valor atual
    ADD #1    ; Incrementa o valor
    STA TEMP  ; Armazena temporariamente
    SUB #4    ; Verifica se chegou a 4
    JNZ LOOP  ; Se não for 4, continua o laço
    HLT       ; Termina o programa
ORG 100
TEMP: DS 1    ; Variável temporária
END 0
```