UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA (CCET)
DEPARTAMENTO DE COMPUTAÇÃO

Microprocessadores e Microcontroladores

Prof. Edilson R. R. Kato

Simulação do computador SAP-1

Autores:
Lucas Eduardo Ragni (RA 414662)
Pedro Felipe Bellini de Campos (RA 414859)
Rafael Joseph Pagliuca dos Santos (RA 414557)

São Carlos, 2013.

Resumo

Neste projeto foi implementado um computador do tipo SAP-1 (Simple as Possible 1) utilizando o software de simulação Proteus. Verificou-se o funcionamento de um computador básico, capaz de somar, subtrair e exibir o resultado através de um conjunto de LEDs.

Objetivos

Neste projeto pretende-se implementar em um ambiente simulacional um computador do tipo SAP-1 (Simple as Possible 1), de forma a compreender o papel desempenhado por seus principais componentes e entender a essência de um sistema capaz de seguir, automaticamente, um dado conjunto de instruções.

Introdução

O computador do tipo SAP-1 segue uma arquitetura projetada para fins acadêmicos, de modo que sua simplicidade e inteligibilidade são alcançadas ao custo de um reduzido número de instruções e de sua baixa velocidade de processamento.

O SAP-1 não pode ser considerado como Turing completo, já que não possui a capacidade de produzir um desvio condicional a partir do algoritmo que está sendo executado, além de outras limitações.

Entre suas principais vantagens estão justamente o número reduzido de instruções, que o torna uma ferramenta didática capaz de exemplificar o funcionamento de um sistema básico de computador.

Instruções do SAP-1

Esta arquitetura reduzida foi projetada para ser capaz de efetuar as seguintes operações:

Mnemônico	Descrição
LDA	Carrega um valor da memória principal no acumulador.
ADD	Soma um valor da memória principal com o acumulador.
SUB	Subtrai um valor da memória principal do acumulador.
OUT	Exibe o valor do acumulador nos LEDs de saída.
HLT	Interrompe a execução do programa.

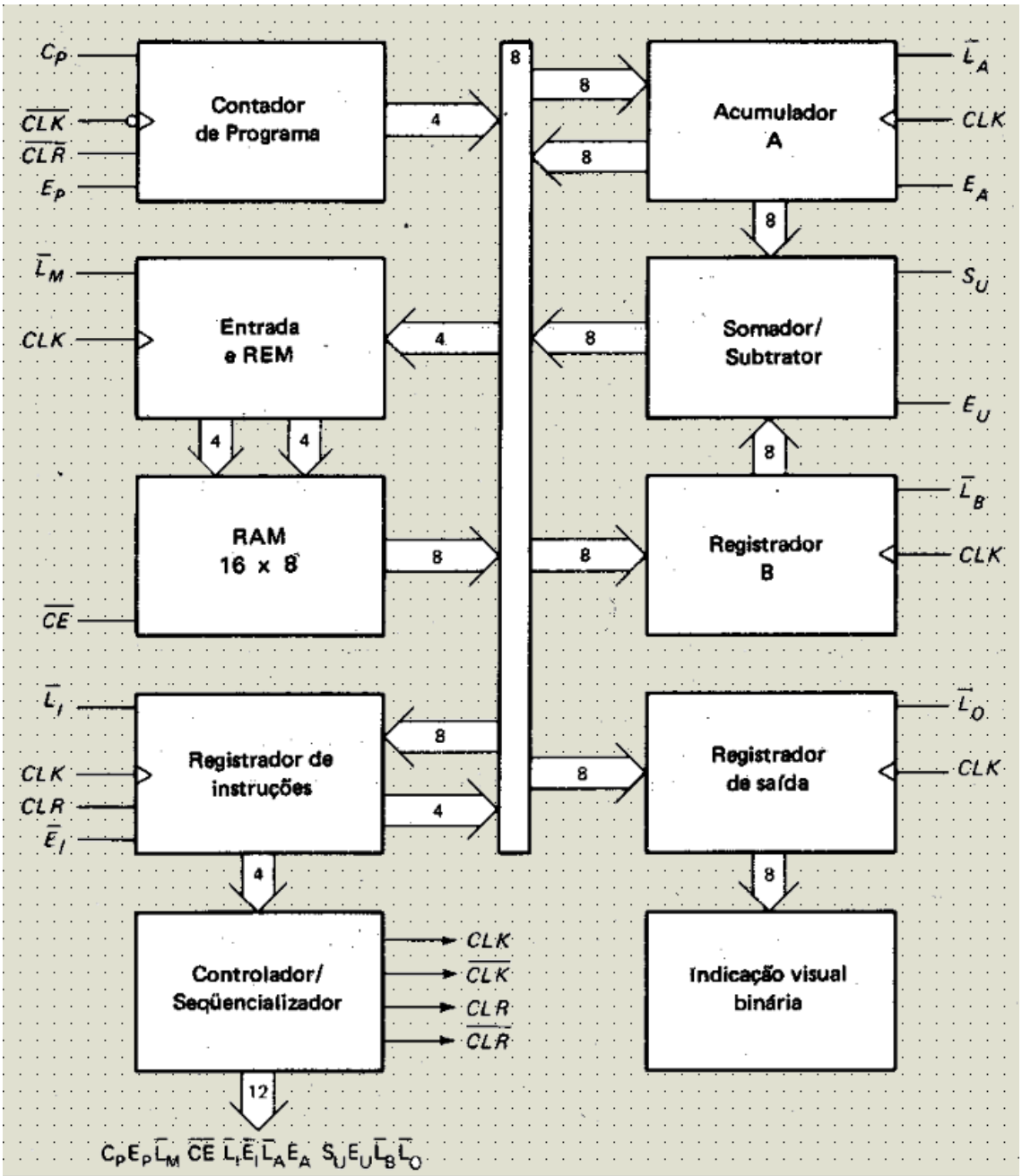
Das cinco instruções, três delas são acompanhadas do endereço do valor a ser utilizado para a operação. Isto é, **LDA**, **ADD** e **SUB** são seguidas de um endereço de 4 bits de onde será lido o dado de 8 bits a ser utilizado.

Já as outras duas instruções, **OUT** e **HLT**, são autosuficientes e não necessitam de qualquer tipo de parâmetro.

Componentes do SAP-1

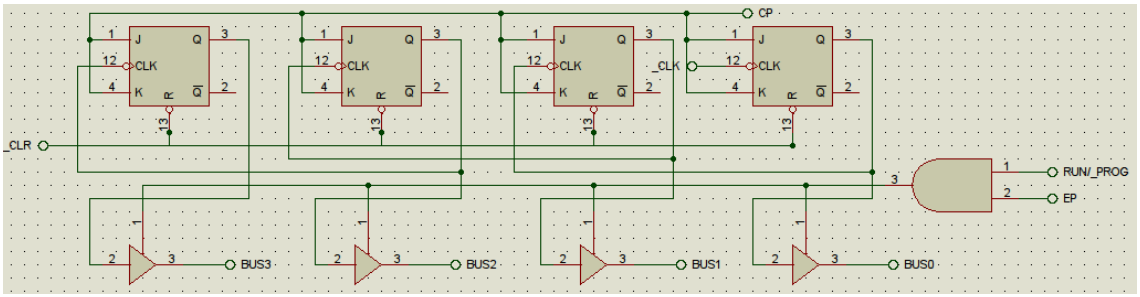
Na referência [2] encontra-se os detalhes técnicos e o diagrama esquemático do circuito eletrônico necessário para a simulação do computador.

A seguir encontra-se um diagrama de blocos que resume os componentes fundamentais do sistema, quais bits de controle interferem sobre cada um deles, e como eles estão conectados entre si:


Contador de Programa
Acumulador A
Entrada e REM
Somador/ Subtrator
RAM 16 x 8
Registrador B
Registrador de instruções
Registrador de saída
Controlador/ Seqüencializador
Indicação visual binária
C_P
$\overline{CLK}$
$\overline{CLR}$
E_P
$\overline{L}_A$
CLK
E_A
$\overline{L}_M$
S_U
E_U
$\overline{CE}$
$\overline{L}_B$
$\overline{L}_I$
$\overline{E}_I$
$\overline{L}_O$
$\overline{CLK}$
CLR
$\overline{CLR}$
8
4
12
$C_P E_P \overline{L}_M \overline{CE} \overline{L}_I \overline{E}_I \overline{L}_A E_A S_U E_U \overline{L}_B \overline{L}_O$

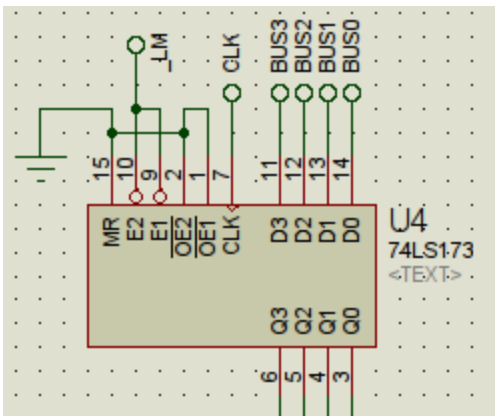
Os componentes fundamentais do computador SAP-1 são:

Contador binário



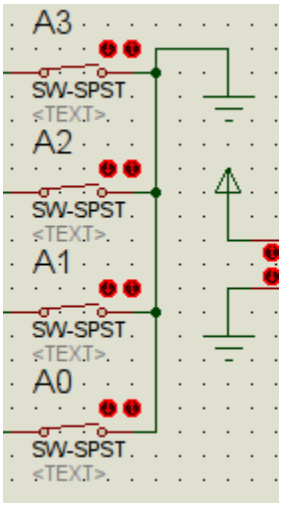
O contador binário inicia a contagem em 0000 assim que a simulação é iniciada, e a cada vez que o bit **CP** da matriz de controle é ativado, tem sua contagem incrementada. Quando o bit de controle **EP** está ativado, a saída do contador ficar disponível em **BUS0**, **BUS1**, **BUS2**, **BUS3**. Desse modo, o computador pode usar o valor da contagem para executar as instruções de forma sequencial.

Latch de instruções



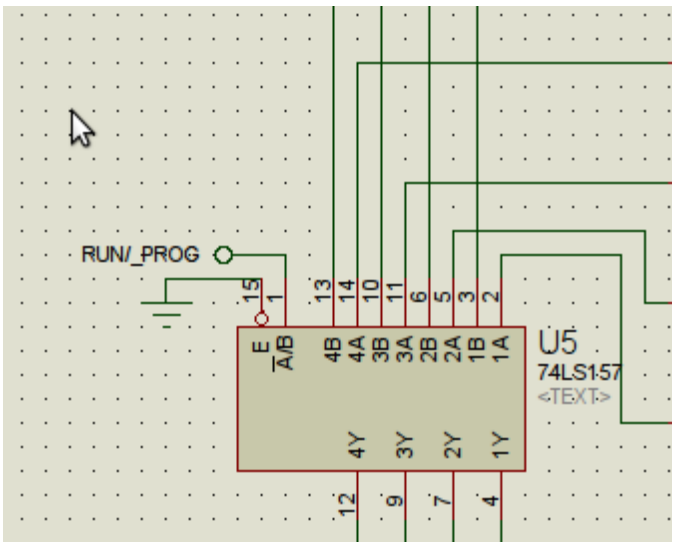
O latch de instruções armazena os 4 bits provenientes de **BUS0**, **BUS1**, **BUS2** e **BUS3** no instante em que o bit de controle **_LM** estiver em nível baixo e houver um pulso do **CLK**. Esses bits podem vir do **contador binário** ou do **latch de endereço do parâmetro da instrução**. A saída desse latch está sempre ativada, e conectada diretamente ao **demultiplexador de endereço**.

Chave de endereço da memória principal



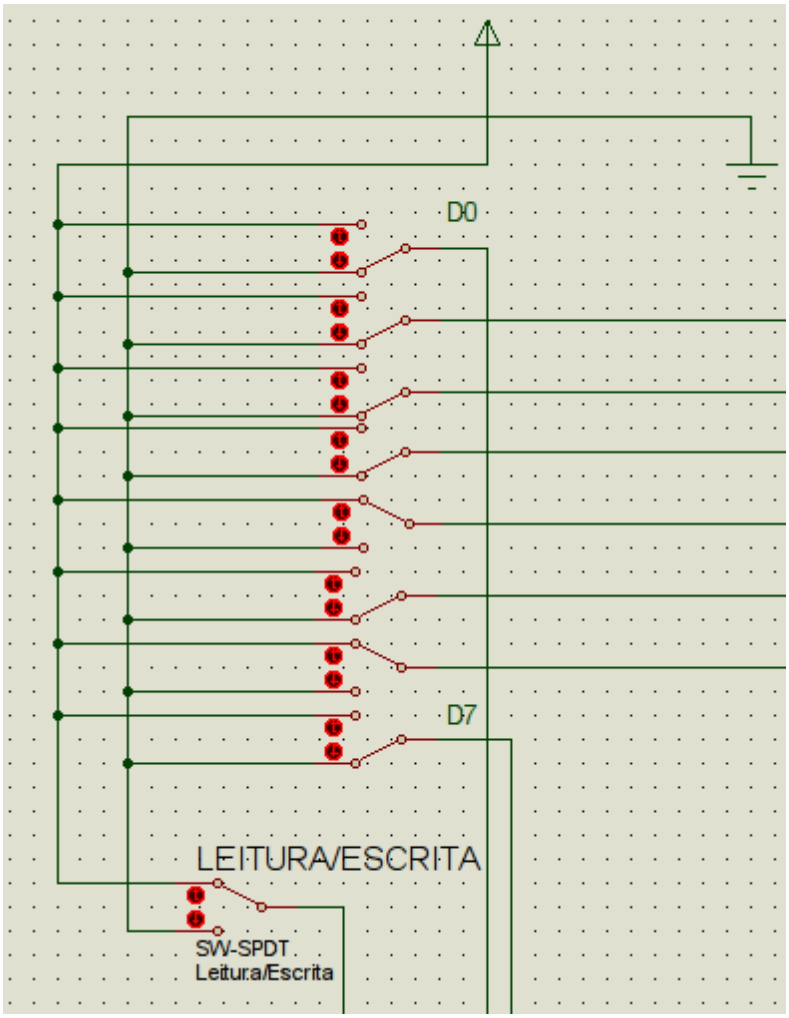
Essas chaves são utilizadas apenas no modo Programação, para definir manualmente o endereço da memória principal que está sendo acessado para escrita.

Demultiplexador de endereço



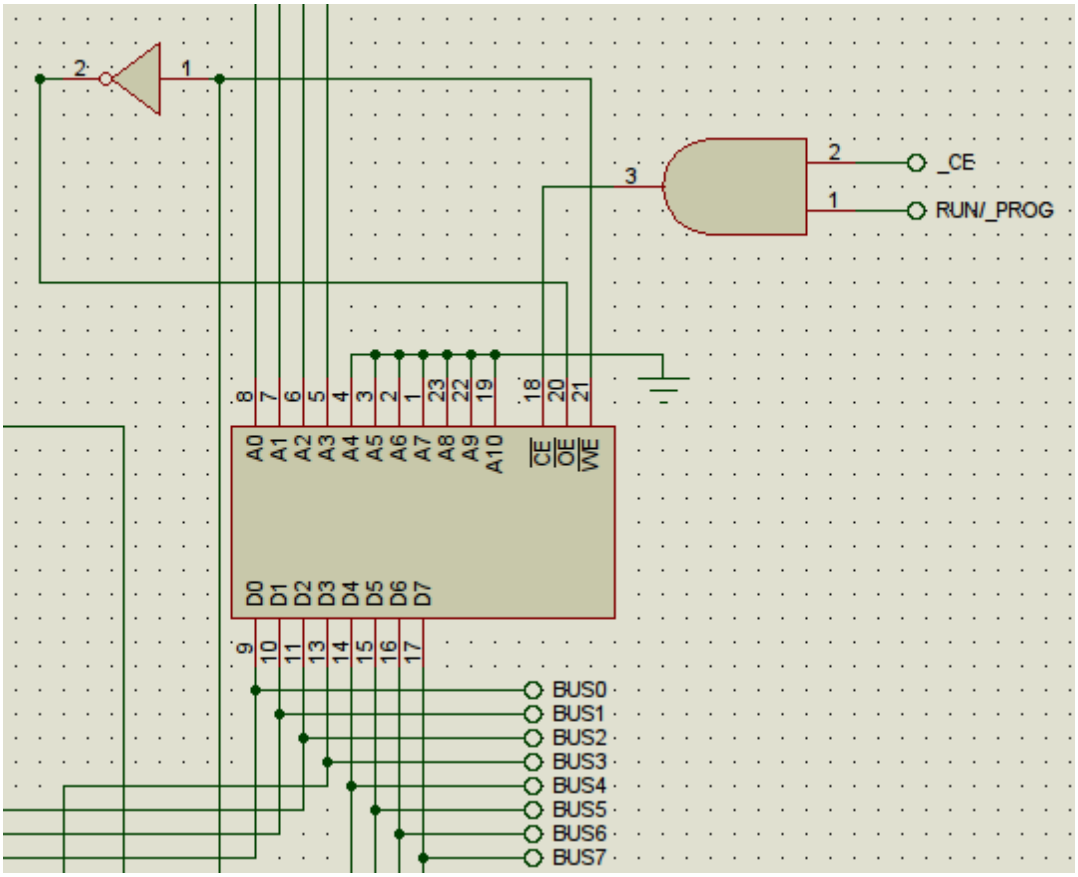
Este componente tem o papel de definir qual dos dois endereços de memória será efetivamente acessado. Essa escolha é feita pelo demultiplexador com base no valor do bit **RUN/_PROG** (se for 0, o endereço é obtido do **latch de endereço**, se for 1, vem das **chaves de endereço**).

Chave de dados da memória principal



A chave de dados da memória principal é o dispositivo usado para programar o SAP. Uma vez escolhida a posição de memória, feita pelas chaves de endereço e o demultiplexador usa-se os bits **D0**, **D1**, **D2**, **D3**, **D4**, **D5**, **D6** e **D7** para determinar os dados, sendo que algumas linhas possuem valores específicos nos 4 bits mais significativos que buscam uma função (por exemplo: LDA = 0000 ; ADD = 0001 ; OUT = 1110 ; HLT = 1111) pré definida pela matriz de controle.

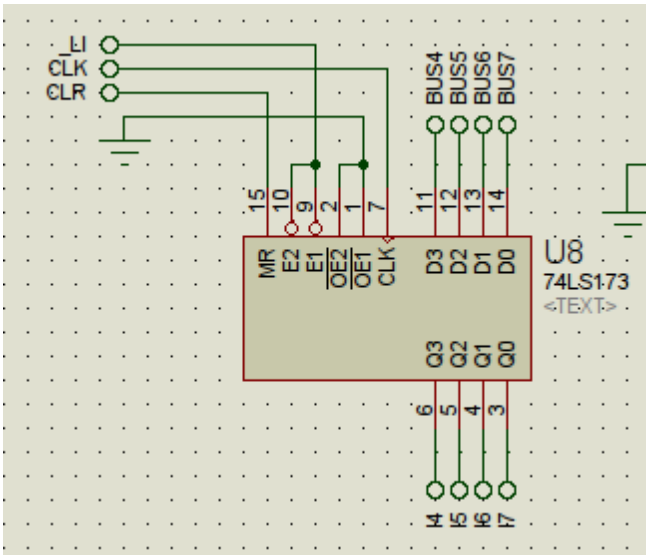
Memória principal (instruções e dados)



A memória principal armazena tanto as instruções quanto os dados que serão utilizados durante o processamento do algoritmo. Quando o bit **RUN/_PROG** encontra-se em nível alto (ou seja, em modo de execução), a placa é ativada sempre que o bit de controle **_CE** estiver em nível baixo. Além disso, o dado armazenado em uma determinada posição de memória pode ser enviado para as saídas **BUS0**-**BUS7**, desde que o bit de controle **_OE** esteja em nível baixo.

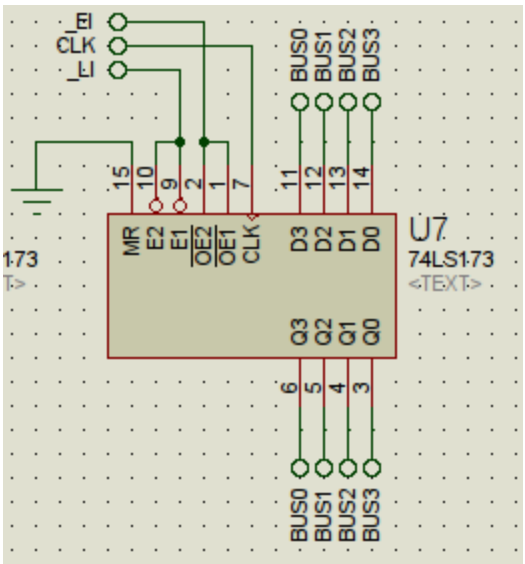
Quando em modo simulação (**RUN/_PROG** em nível baixo), a placa está sempre ativada, e pronta para receber os dados provenientes das **chaves de programação.**

Latch de instrução



Este latch armazena o código da instrução que está sendo atualmente executada (ou seja, os 4 bits mais significativos salvos em um determinada posição de memória), assim que o bit de controle **_LI** estiver em nível baixo e tiver um pulso em **CLK**. As saídas **Q0-Q4** estão sempre ativadas e conectadas diretamente ao **decodificador de instruções**.

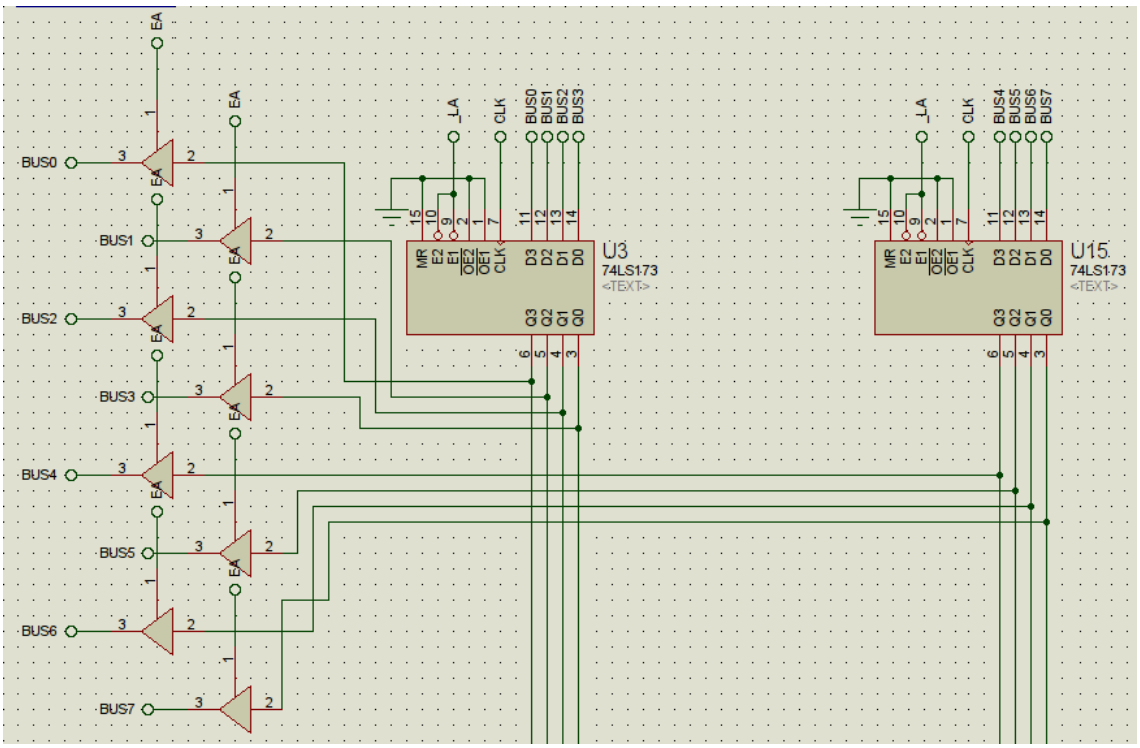
Latch de endereço do parâmetro da instrução



Este latch armazena os 4 bits menos significativos lidos da posição de memória principal sendo acessada em um determinado instante, que equivalem ao endereço do parâmetro das instruções **LDA**, **ADD** e **SUB**. Assim como o **latch de instruções**, o dado é atualizado quando o bit de controle **_LI** está em nível baixo.

Sua saída é tristate controlada pelo bit **_EI**, que quando em nível baixo permite a passagem do endereço através dos fios **BUS0-BUS4**, que será lido pelo **latch de endereço**.

Latches do valor do acumulador

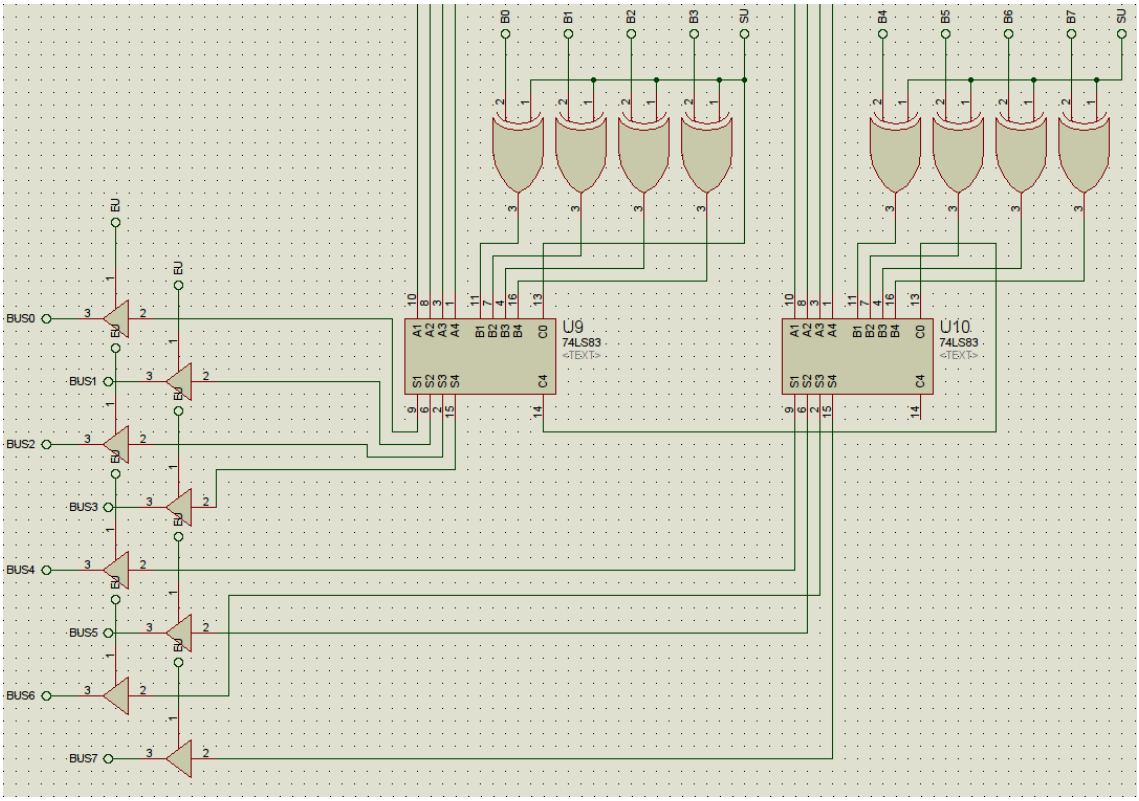


Cada um dos **latches do acumulador** usado na figura acima possui apenas quatro bits, e como os dados da memória usam oito bits foi necessário a implementação de dois chips. Sua função é receber os dados do barramento (**BUS0** até **BUS7**) e guardá-lo.

Uma vez que os dados estejam no acumulador eles serão disponíveis automaticamente no circuito somador/subtrator.

Porém, nem sempre é interessante que eles sejam encaminhados ao barramento, e por isso faz-se uso das portas tristate. Se elas forem acionadas pelo bit de controle **EA** o valor do acumulador é encaminhado ao barramento, caso contrário o barramento permanece livre.

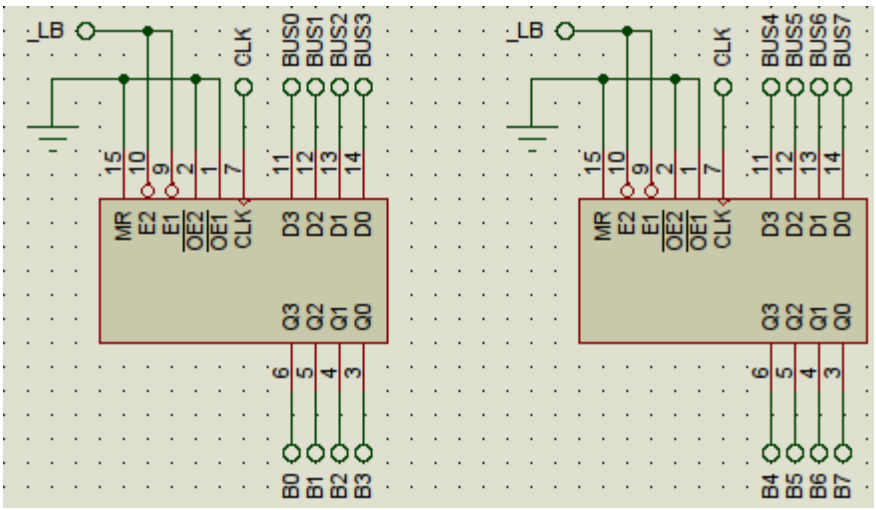
Somador e subtrator completo



Esta parte tem o papel de possibilitar a soma e subtração de um valor qualquer **B** com o valor do **acumulador**. O somador e subtrator é um circuito combinatório independente do clock do circuito, mas sua saída é controlada através de portas tristate ativadas pelo bit de controle **EU.**

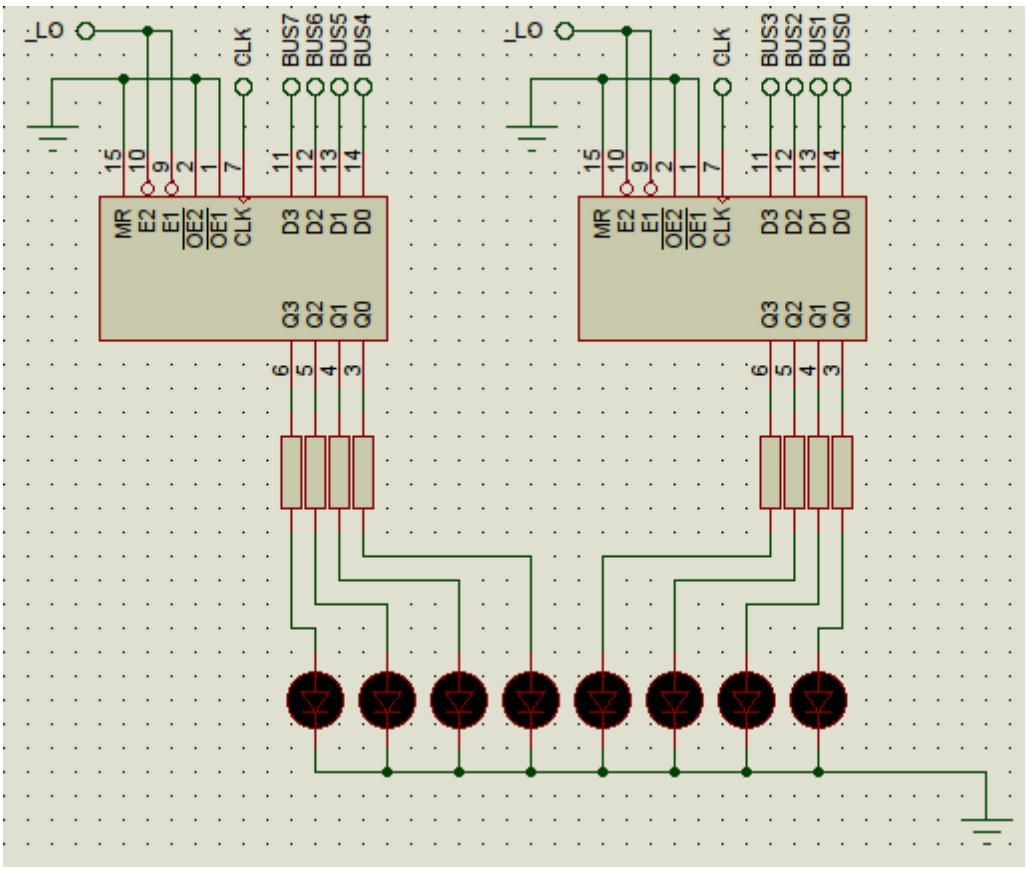
Sua saída é enviada para os fios **BUS0-BUS7**, para poderem ser carregadas nos **latches do acumulador**, de forma que o resultado da operação substitua automaticamente o valor original do **acumulador**.

Latch do valor de B



Estes latches armazenam um valor qualquer **B** que é lido da memória principal pelas instruções **ADD** e **SUB**. A saída desses latches está conectada ao circuito somador/subtrator.

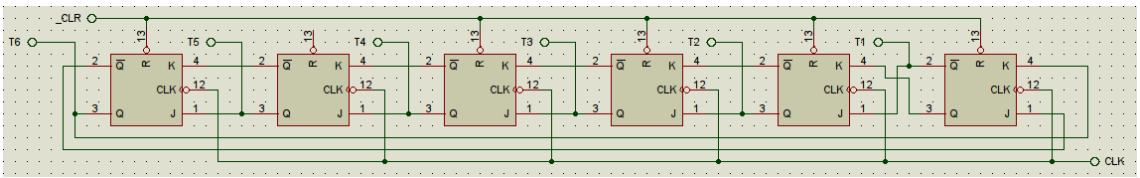
Latches da saída (ligados aos LEDs)



Os **latches de saída** são acionados atráves do bit de controle **_LO**. Ao se acionar

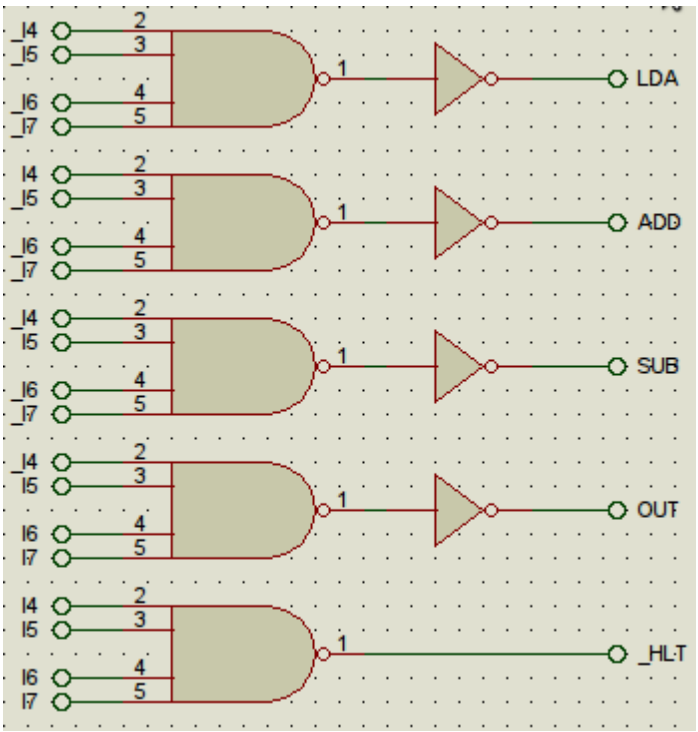
esse bit, o valor que estiver no barramento é transferido aos latches que automaticamente (devido aos pinos **_OE1** e **_OE2** estarem aterrados) acionam os respectivos LEDs a fim de imprimir o valor esperado para cada programa em particular.

Contador em anel



O contador em anel é um conjunto de flip-flops do tipo JK que tem como função sequencializar a ordem de envio dos bits de controle. A cada pulso do clock, o flip-flop que estiver em nível alto define a etapa de processamento atual (de **T1** a **T6**). Por ser um contador em anel, apenas um deles estará em nível alto por vez, e a rotação se dá de forma cíclica.

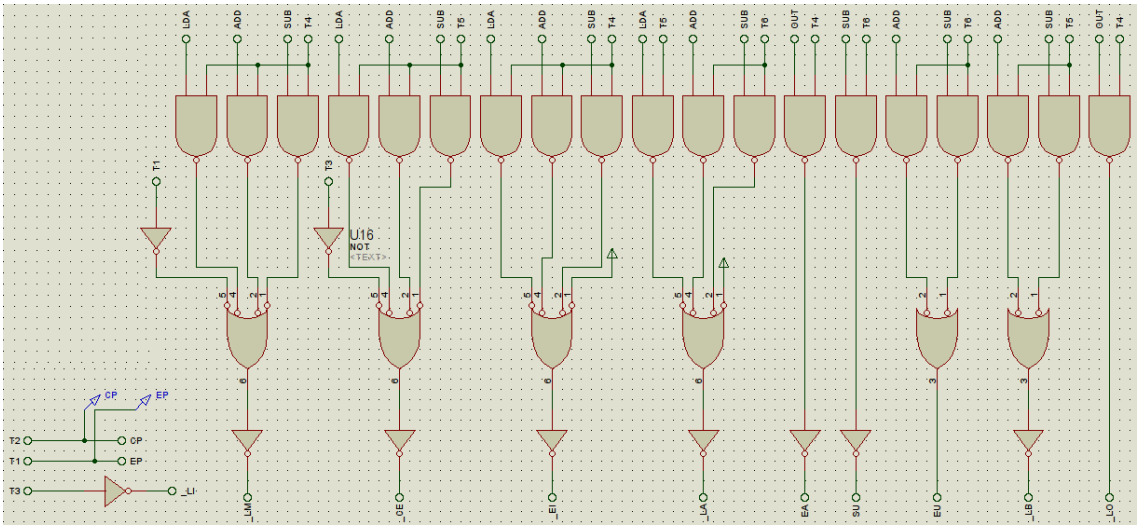
Decodificador de instruções



O **decodificador de instruções** é responsável por ativar um dos bits **LDA**, **ADD**, **SUB**, **OUT** ou **_HLT** dependendo da configuração dos bits **I4**, **I5**, **I6** e **I7.** Dessa forma, um código arbitrário de instrução é associado a uma instrução específica (por exemplo, o estado {I7, I6, I5, I4} = {0, 0, 0, 0} ativa o bit **LDA**). Posteriormente, os bits relativos a cada instrução são utilizados pela matriz de controle para formar a **palavra de controle**,

juntamente com o número da etapa atual fornecido pelo **contador em anel**.

Matriz de controle



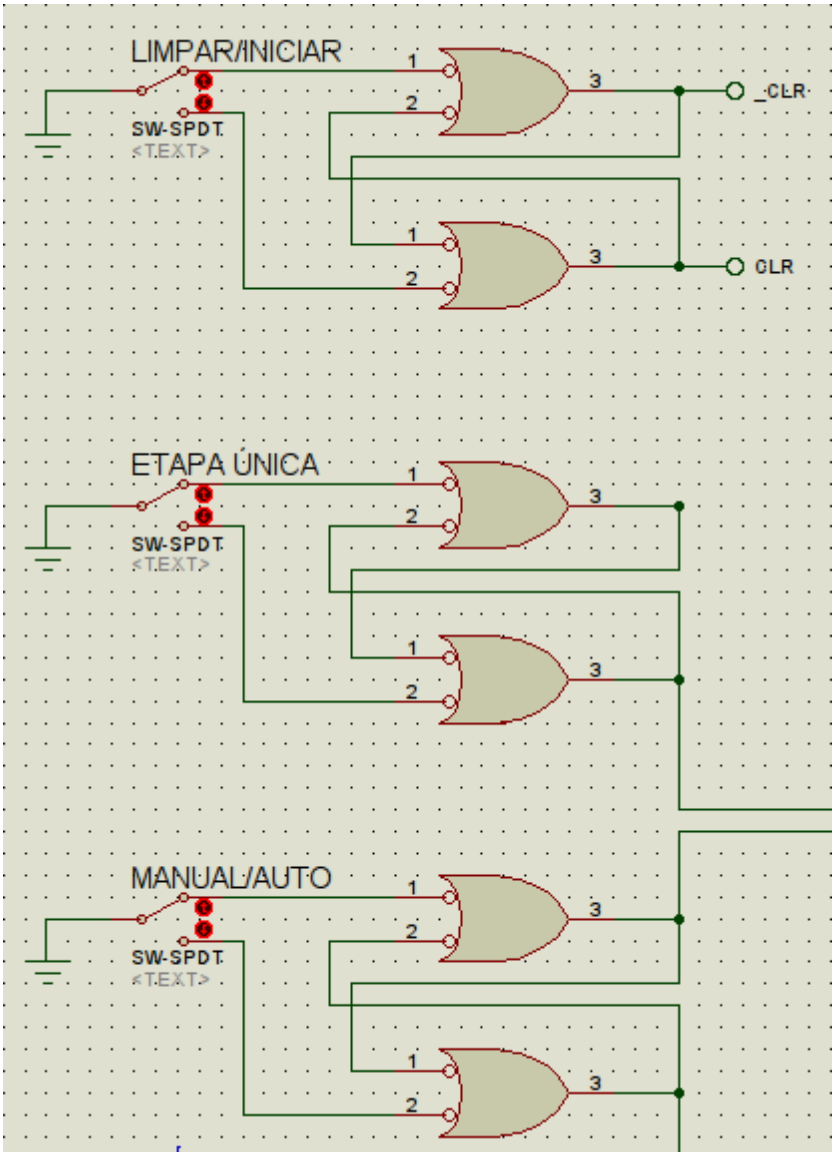
A matriz de controle recebe do **decodificador de instrução** qual instrução está sendo atualmente executada, e recebe do **contador em anel** qual é a etapa atual.

Utilizando essas duas informações, ela forma a **palavra de controle** e envia para o restante do circuito. Ou seja, para uma etapa de execução e para uma instrução específica, um determinado conjunto de ações devem ser executadas. Por exemplo: na instrução **ADD**, durante a **etapa 4**, o campo de instruções é enviado para o decodificador de instruções e o campo de endereço do parâmetro da instrução é enviado para o **latch de instrução**, o que necessita que os bits de controle **_EI** e **_LM** estejam ativados.

Isso resulta na seguinte palavra de controle:

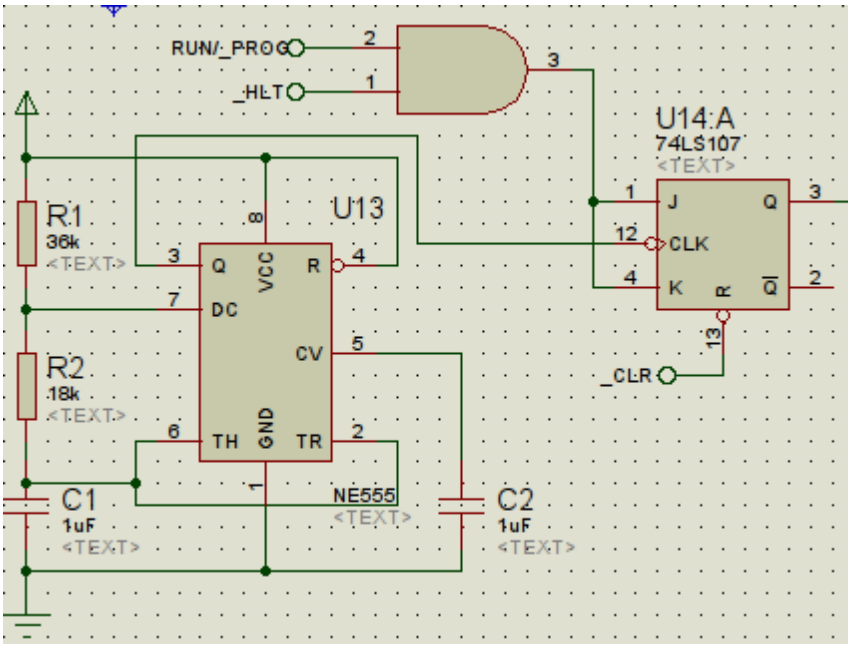
CP	EP	**_LM**	_CE	_LI	**_EI**	_LA	EA	SU	EU	_LB	_LO
0	0	**0**	1	1	**0**	1	0	0	0	1	1

Chaves de controle de execução



Essas chaves são necessárias para que o usuário que esteja simulando o computador SAP-1 possa reiniciar o processamento, escolher entre modo de execução automática ou manual (passo a passo), e, caso em modo manual, avançar uma etapa.

Gerador do clock



Esta parte é responsável por gerar os pulsos de clock do circuito, utilizando como base um temporizador **555**. Os valores de capacitância e resistência diferem um pouco do projeto original disponível em [1], para minimizar a carga de processamento utilizada pelo Proteus para simular o circuito.

Programando o SAP-1

O computador SAP-1 é programado a partir das **chaves de endereço** e **chaves de dados** da memória principal, juntamente com a chave **RUN/_PROG**.

Através dessas chaves, o usuário tem a sua disposição 16 posições na memória principal que podem armazenar dados de até 8 bits.

O computador irá começar na posição **0H** da memória e seguirá até a posição **FH**, exceto se a instrução **HLT** for detectada. Para cada posição de memória, será lido o byte correspondente:

Estrutura do byte lido: **I7 I6 I5 I4 P3 P2 P1 P0**.

Os 4 bits mais significativos (**I7**, **I6**, **I5**, **I4**) do byte correspondem a um **código de instrução**. Os 4 bits menos significativos (**P3**, **P2**, **P1**, **P0**) informam o endereço da memória principal de onde se encontra o dado que será utilizado como parâmetro da instrução. No caso das instruções que dispensam o uso de parâmetro, os 4 bits menos significativos são do tipo “don’t-care”.

Segue lista associando os mnemônicos das instruções do SAP-1 com seus

respectivos códigos de instrução:

Mnemônico	Código da instrução (binário)	Código da instrução (hexa)
LDA	0000	0H
ADD	0001	1H
SUB	0010	2H
OUT	1110	EH
HLT	1111	FH

Exemplo de rotina

Para executar uma rotina em que o computador realize a operação 1 + 3 + 5 + 7 + 9 e exiba o resultado 25 através dos 8 LEDs de saída, é necessário que a memória principal seja alimentada da seguinte forma:

Endereço de memória	Dado da memória	Instrução	Descrição
0000	0000 1010	LDA AH	Carrega o valor da posição 1010 (valor 1) no acumulador.
0001	0001 1011	ADD BH	Soma o valor da posição 1011 (valor 3) ao valor do acumulador, e guarda o resultado no acumulador.
0010	0001 1100	ADD CH	Soma o valor da posição 1100 (valor 5) ao valor do acumulador, e guarda o resultado no acumulador.
0011	0001 1101	ADD DH	Soma o valor da posição 1101 (valor 7) ao valor do acumulador, e guarda o resultado no acumulador.
0100	0001 1110	ADD EH	Soma o valor da posição 1110 (valor 9) ao valor do acumulador, e guarda o resultado no acumulador.
0101	1110 XXXX	OUT	Exibe o resultado do acumulador através dos LEDs de saída.
0110	1111 XXXX	HLT	Interrompe a execução do programa.
- - -			

1010	0000 0001	Valor 1	Valor a ser utilizado em uma das operações.
1011	0000 0011	Valor 3	Valor a ser utilizado em uma das operações.
1100	0000 0101	Valor 5	Valor a ser utilizado em uma das operações.
1101	0000 0111	Valor 7	Valor a ser utilizado em uma das operações.
1110	0000 1001	Valor 9	Valor a ser utilizado em uma das operações.

Observa-se no programa acima que todas as posições de memória após a instrução **HLT** podem ser utilizadas para armazenar valores numéricos, já que o computador não irá interpretar as mesmas como instruções. Para fins de legibilidade, no entanto, optou-se por separar a posição de memória **AH** e as seguintes (até **EH**) para armazenar os dados, formando um gap na memória.

Melhorias em relação ao projeto original

O ato de programar manualmente a rotina do SAP-1 é uma tarefa tediosa e muito sujeita a erros, visto que, para executar um programa simples, com por exemplo 4 instruções e 2 dados, é necessária a configuração de **72 chaves diferentes** (4 chaves de endereço para cada um dos 6 endereços de memória = 24 chaves; 8 chaves de endereço para cada um dos 6 endereços de memória = 48 chaves). Se qualquer uma dessas 72 chaves for configurada de maneira errada, o programa não terá o comportamento esperado.

Sabendo disso, o projeto original foi adaptado para conter uma **EPROM** (código do componente 27256) no lugar da **RAM** para ser utilizada como memória principal do computador. Esta **EPROM** pode ser configurada no ambiente Proteus para ser carregada a partir de um arquivo no formato **Intel Hex** (vide referência [3]).

Além disso, foi desenvolvido um pequeno script utilizando a linguagem PHP para transformar um código-fonte escrito com os mnemônicos do SAP-1 no seu correspondente **Intel Hex**.

Exemplo de código-fonte antes da compilação:

```
; Instruções
0: LDA A
1: ADD B
```

```
2: OUT
3: HLT

; Dados
A: 4
B: 6
```

Ao processar o código-fonte acima, o compilador ignora as linhas iniciadas com ponto e vírgula (;). Toda linha obrigatoriamente deve ser iniciada com a posição de memória correspondente seguida por dois pontos (:).

Resultado (arquivo **Intel Hex**) após a compilação:

```
:020000040000FA
:040000000A1BE0F007
:02000A000406EA
:00000001FF
```

Esse arquivo pode ser facilmente configurado para alimentar a memória **EPROM**, fazendo com que a programação seja mais rápida e menos suscetível a erros, além de se tornar mais próxima à realidade de programação de microprocessadores e microcontroladores, tornando-se uma introdução à programação destes.

O projeto adaptado com a **EPROM** e o respectivo **compilador** do SAP-1 estão disponíveis na seguinte URL:
http://rs.anoluz.net/2013/10/simulando-um-computador-sap-1-simple-as-possible-1/

Conclusões

Ao fim desse projeto, pode-se entender como é o funcionamento básico de um computador bem simples, capaz de efetuar poucas operações como LDA, ADD, SUB, OUT e HLT. É interessante ressaltar que é relativamente fácil acrescentar outras funções a esse computador.

Para isso, seria necessário adicionar uma nova porta NAND ao conjunto **decodificador de instrução**, além de planejar quais bits de controle deveriam ser ativados em cada uma das etapas **T4**, **T5** e **T6** (visto que as primeiras etapas **T1**, **T2**, **T3** são comum a todas as instruções) e efetuar essas alterações através de respectivas adaptações na matriz de controle.

Bibliografia

[1] Autor desconhecido. **Descrição da Arquitetura SAP-1**. Documento digital disponível em http://www.ic.unicamp.br/~ducatte/mc542/2012S2/sap-1.pdf. Acesso em 20 de outubro de 2013.

[2] STALLINGS, W. **Computer Organization and Architecture - Designing for Performance**. 8ª edição. Editora Pearson Education, Inc. Nova Jersey, 2010.

[3] Intel Hexadecimal Object File Format Specification. Documento digital disponível em http://www.interlog.com/~speff/usefulinfo/Hexfrmt.pdf. Acesso em 20 de outubro de 2103.