

CONSTRUÇÃO DE UMA UCP HIPOTÉTICA

M++

INTRODUÇÃO

O seguinte artigo apresenta uma UCP hipotética construída no software simulador DEMOWARE Digital Works 3.04.39. A UCP (Unidade Central de Processamento) é capaz de executar instruções lógicas e aritméticas, movimentação de dados envolvendo memória e registrador, chamadas de subrotinas e E/S (Entrada e Saída). O objetivo é mostrar o funcionamento básico da mesma, podendo ser visualizado sua execução passo e ao mesmo tempo, visualizar os sinais gerados por cada instrução, quando aplicado o *clock*. O presente também demonstra o funcionamento de um montador *assembly* que será responsável pela geração do código de máquina para a UCP hipotética.

A idéia da UCP hipotética foi apresentada pelo aluno do curso de Ciências da Computação, Jonathan Borges, o qual a desenvolveu como trabalho final da disciplina.

DESENVOLVIMENTO

Para entender o funcionamento da UCP hipotética, primeiramente devemos conhecer o SIMULADOR onde a mesma será simulada. A figura 1 mostra a tela de execução.

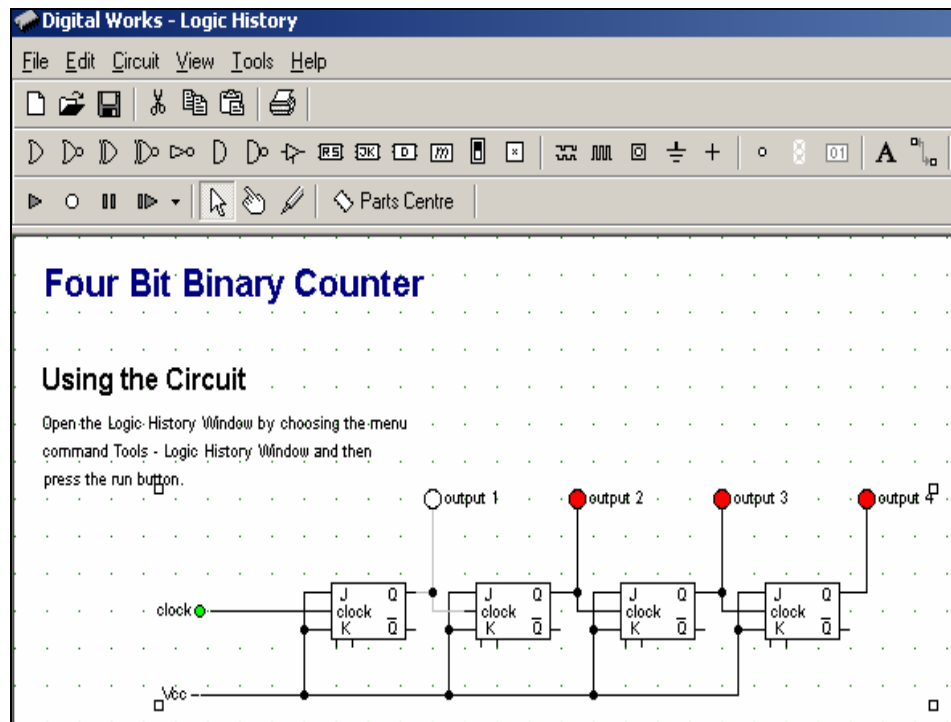


Figura 6 – Exemplo de um contador de 4 bits

A UCP hipotética é formada basicamente pelos seguintes elementos: ULA (Unidade Lógica Aritmética), acumulador, banco de registradores, memória de controle, barramento de dados, barramento de controle, barramento de endereço, memória ROM e memória RAM. Cada elemento será descrito posteriormente.

A figura ilustra a visão externa da UCP (macro), a qual possui toda lógica de seu funcionamento. O Digital Works permite a criação de macros.

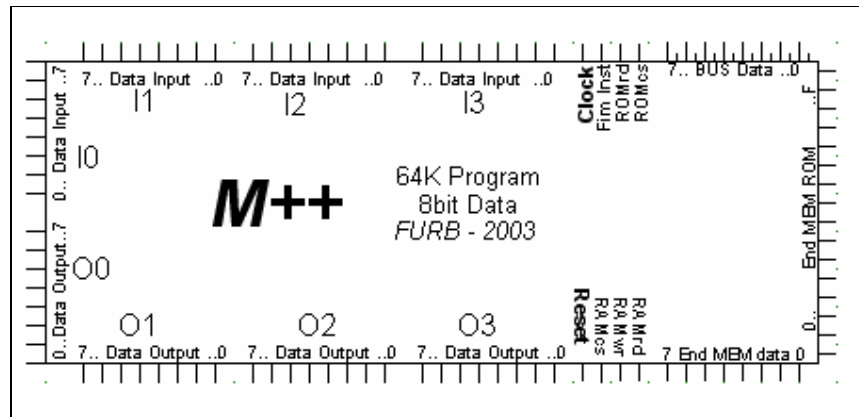


Figura 7 – Imagem externa da UCP (visão MACRO)

Já a figura 8 mostra a UCP conectada a uma memória de programa ROM e também a RAM.

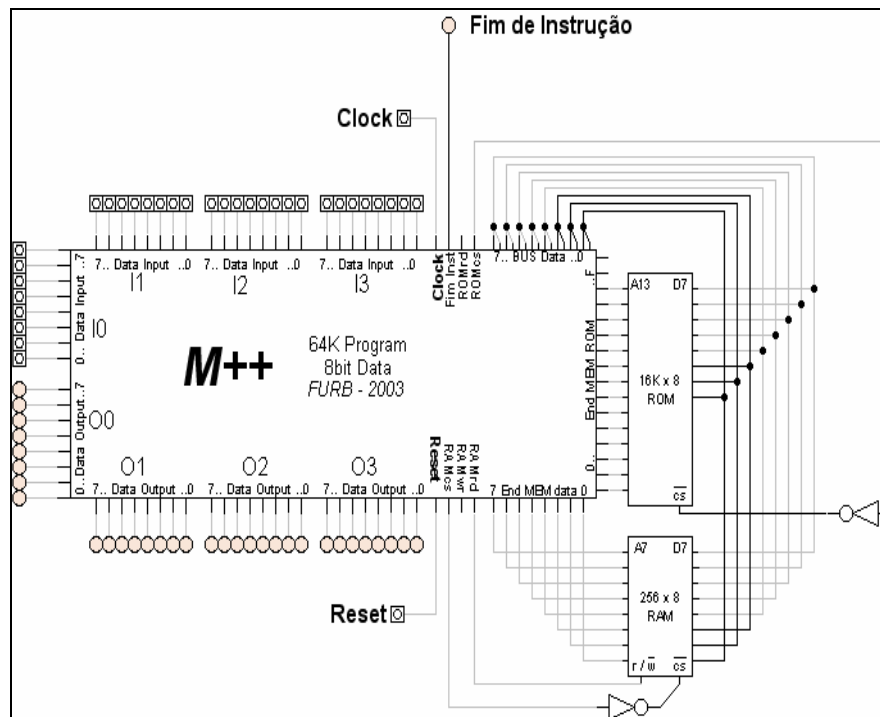


Figura 9 – UCP conectada com memora ROM e RAM

Caraterísticas da UCP

- 4 portas de entrada (I0 e I3);
- 4 portas de saída (O0 a O3);
- Entrada de sinais de *Clock* e *Reset*;
- Indicativo de final de execução de uma instrução;
- Controle da Memória de Dados e Pilha;
- Controle da Memória do Programa;
- Barramento de endereço da Memória de Dados e Pilha (256 - 8bit);
- Barramento de endereço da Memória de Programa (64k - 16bit);
- Barramento de Dados de 8 bits.

Visão interna da UCP hipotética

Clicando na UCP (macro), você poderá ver o que há dentro dela. Macros podem ser incluídas dentro de macros. Veja a figura 10.

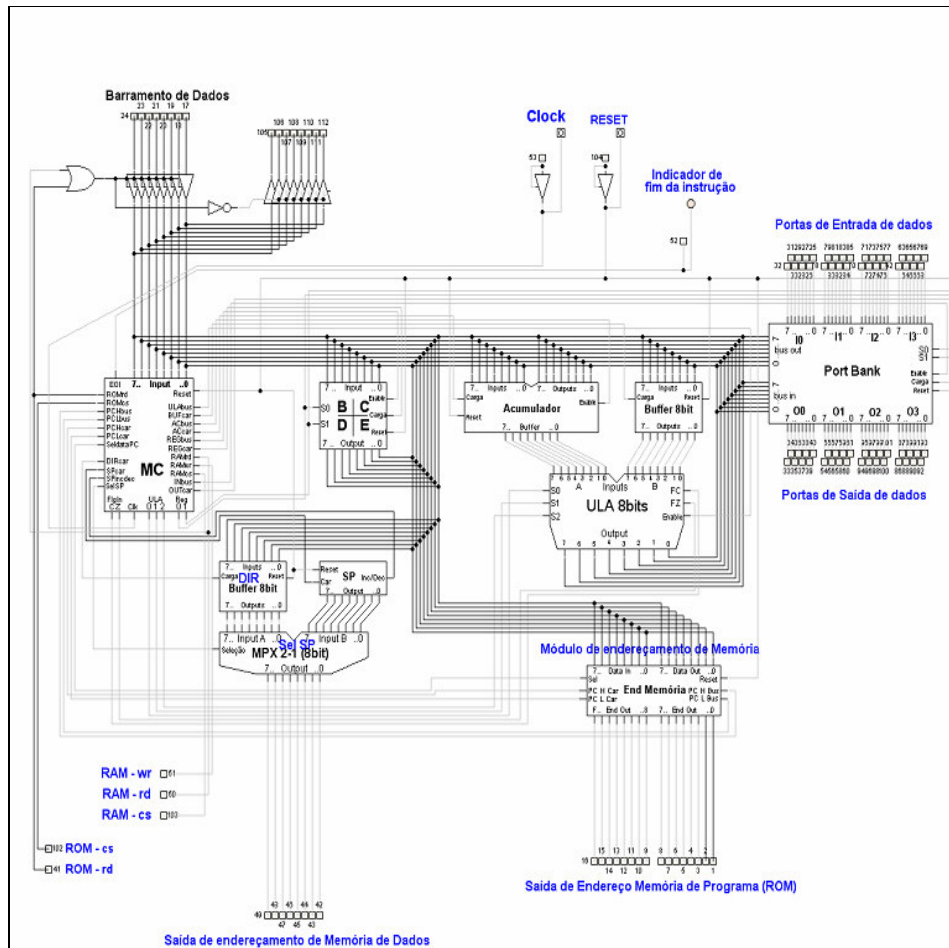


Figura 11 – Visão interna da UCP

A seguir, será apresentado o que faz cada um dos itens que compõem a UCP (macro).

ULA

Realiza as operações lógicas e aritméticas que são: soma, subtração, *and*, *or*, *xor*, *not* e incremento. Possui 2 *flags* que sinalizam se a última operação lógica aritmética resultou em zero ou estouro de operação (*overflow*) (FZ e FC). Veja figura 12, observe os blocos que fazem as operações lógicas e aritméticas, observe também os multiplexadores que selecionarão quais destas operações serão disponibilizadas em uma saída.

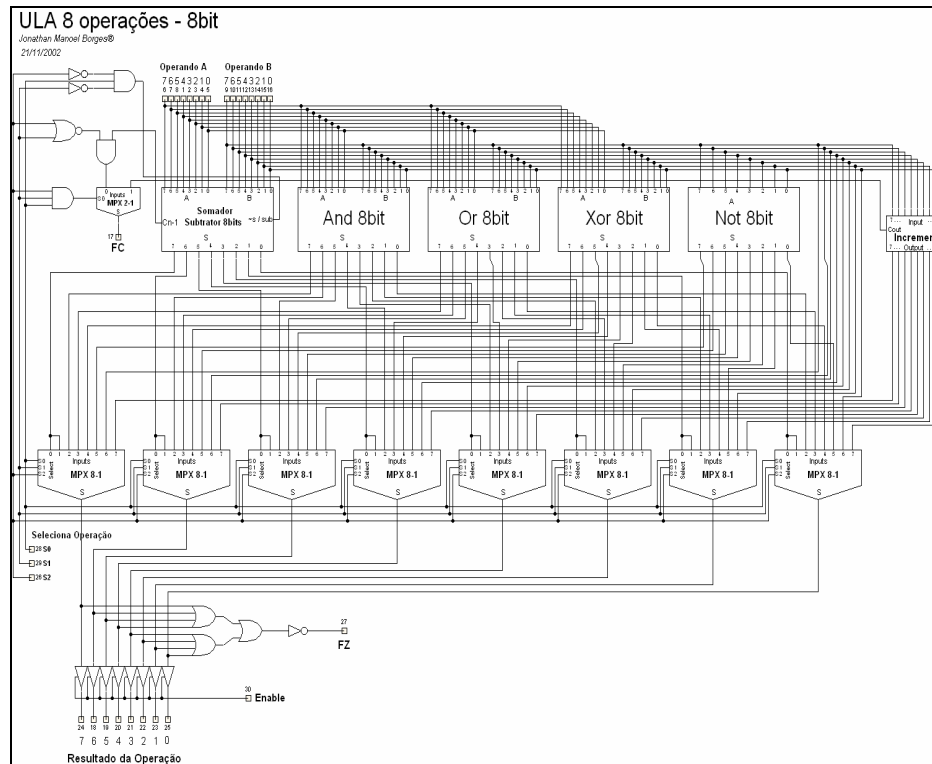


Figura 13 – Visão interna da ULA

Acumulador

O acumulador nada mais é que um conjunto de *flip-flops* tipo D e tem a função de armazenar dados intermediários, utilizados em operações posteriores. A figura 14 ilustra o acumulador. Observe que a saída do acumulador tem circuitos *tristate*, são utilizados para isolá-lo do barramento de dados quando não utilizado.

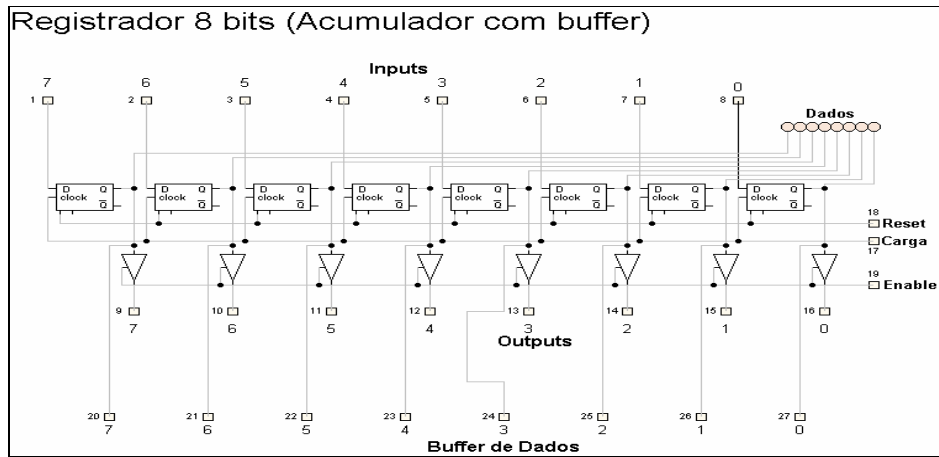


Figura 15 – Visão interna do acumulador

Banco de Registradores

Possui 4 registradores de propósito geral, B,C,D e E, podendo serem utilizados como variáveis. Veja figura 16.

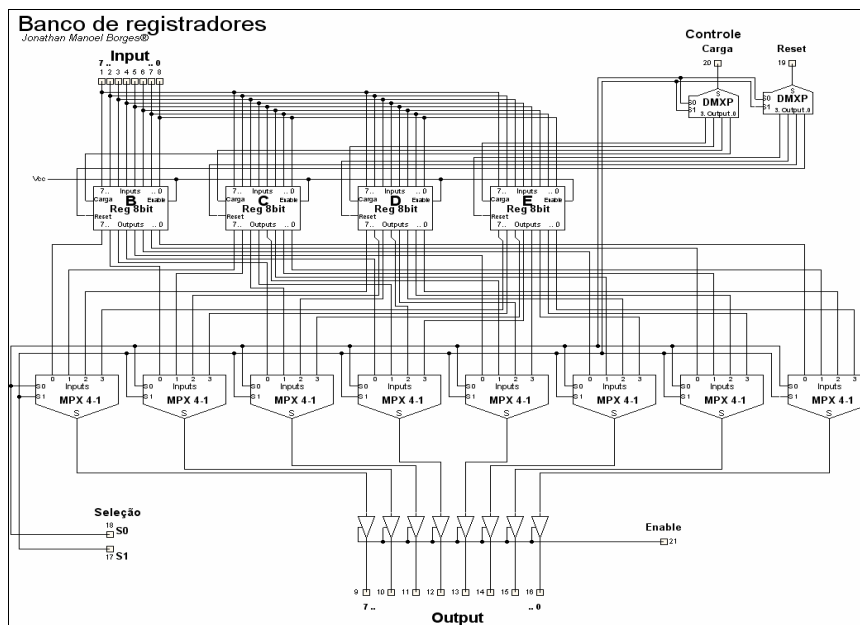


Figura 17 – Visão interna do banco de registradores

Memória RAM

Memória responsável pelo armazenamento de dados temporários do programa. Se a quantidade de registradores do *port bank* não é suficiente para armazenarem todos os seus dados, utilize a memória externa acoplada a UCP. A figura ilustra a memória de dados. A memória pode ser configurada para diferentes tamanhos. Esta memória também é utilizada pela pilha para armazenar o endereço de retorno de subrotinas. Veja a figura 22.

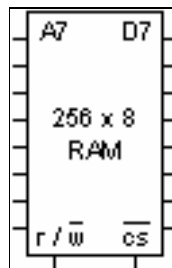


Figura 23 – Memória de dados e pilha

Barramento de Dados

O barramento permite compartilhar dados por todos os componentes, sendo que os mesmos podem ou não entrar em *tristate*. Durante a simulação a sua cor muda.

Barramento de Endereços

Possui dois barramentos de endereço: barramento para a memória de dados e pilha, e para a memória de programa. Durante a simulação a sua cor muda.

Barramento de Controle

Gera todos os sinais de sincronização entre os blocos internos da UCP hipotética.

Módulo de controle

Responsável pela geração de todos os sinais de controle necessários para que a instrução alcance o seu objetivo. A figura 24 ilustra a memória de controle. Observe que dentro do módulo de controle há 2 memórias ROM, são nelas que as seqüências de sinais de controle são armazenados. A Figura 25 ilustra a seqüência de sinais de controle gravadas na ROM. A adição de novas instruções ou blocos envolverá a adição de novos sinais de controle dentro do módulo de controle.

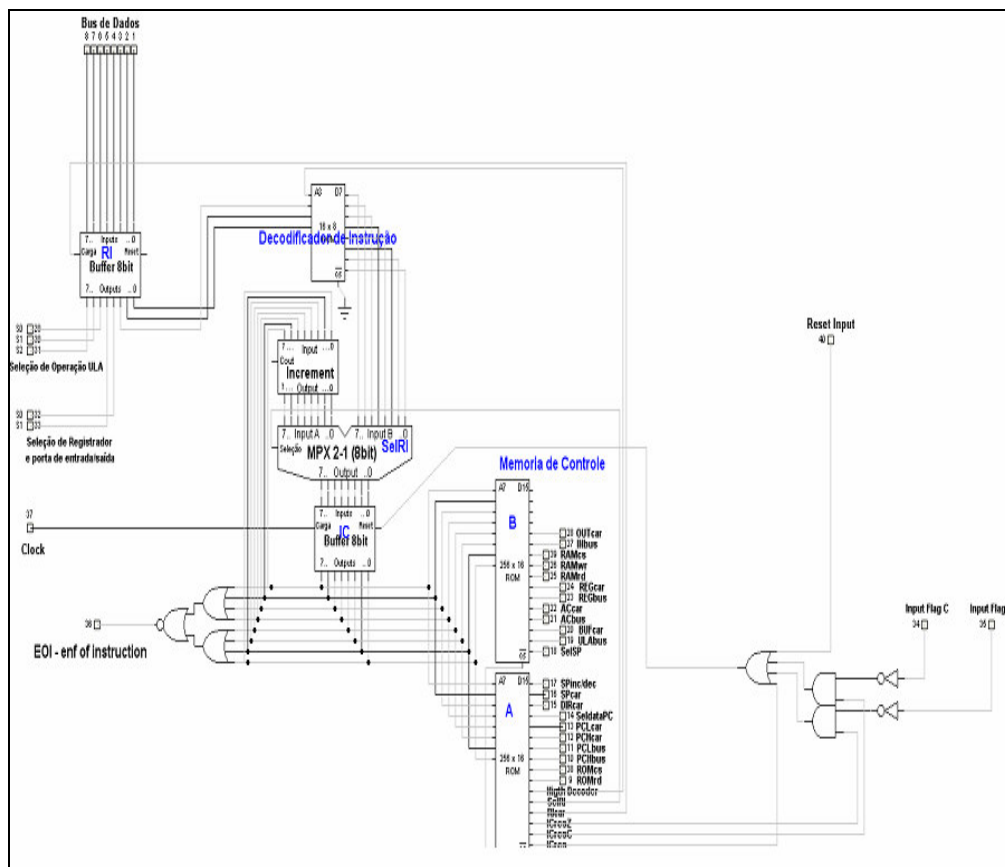


Figura 26 – Módulo de controle

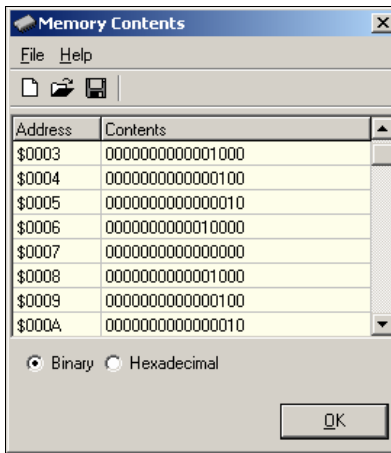


Figura 27 – Conteúdo da memória de controle

Geração de Código de Máquina para UCP hipotética (AssemblyM++)

Para a geração de código de máquina para a UCP, foi construído um montador, permitindo que a UCP possa ser programada com uso de mnemônicos *assembly*. Nas primeiras versões da UCP, a programação era feita em linguagem de máquina. A figura 28 ilustra o montador.

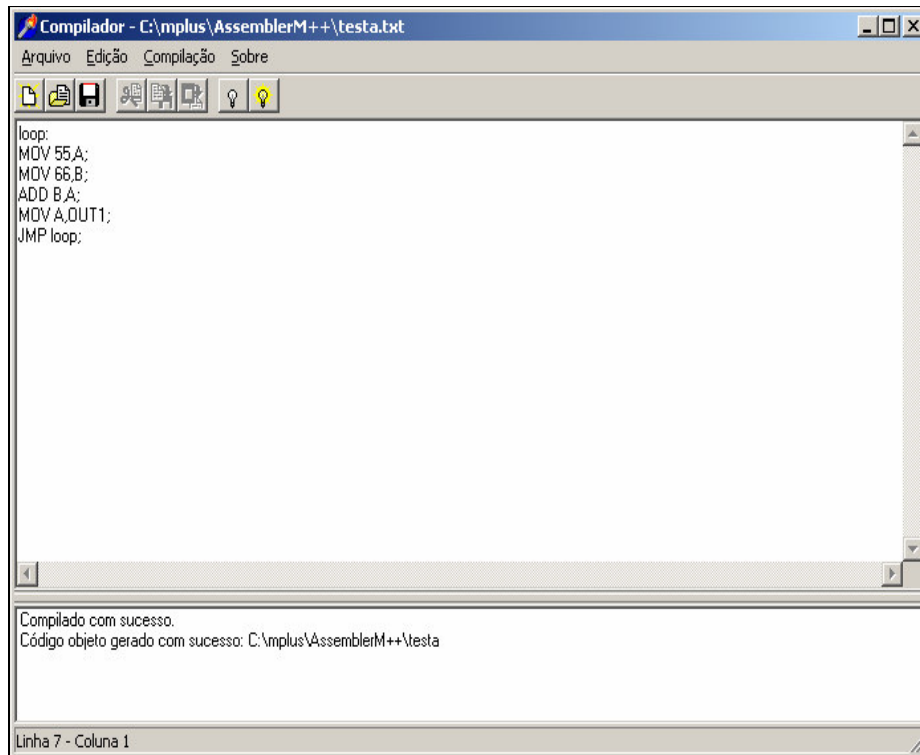



Figura 29 – Imagem do montador assembly

O programa move 55 (hexadecimal) para o acumulador, 66 (hexadecimal) para o registrador B, faz uma soma de B com A colocando o resultado em A e finalmente, move A para a E/S (OUT1).

Carregando o programa na memória ROM

Clicando com o mouse no componente memória ROM (veja figura 30), uma janela abrirá (certifique-se que o ícone  esteja habilitado), solicitando que o usuário entre com o programa em código de máquina gerado pelo montador (terminação .MAP)

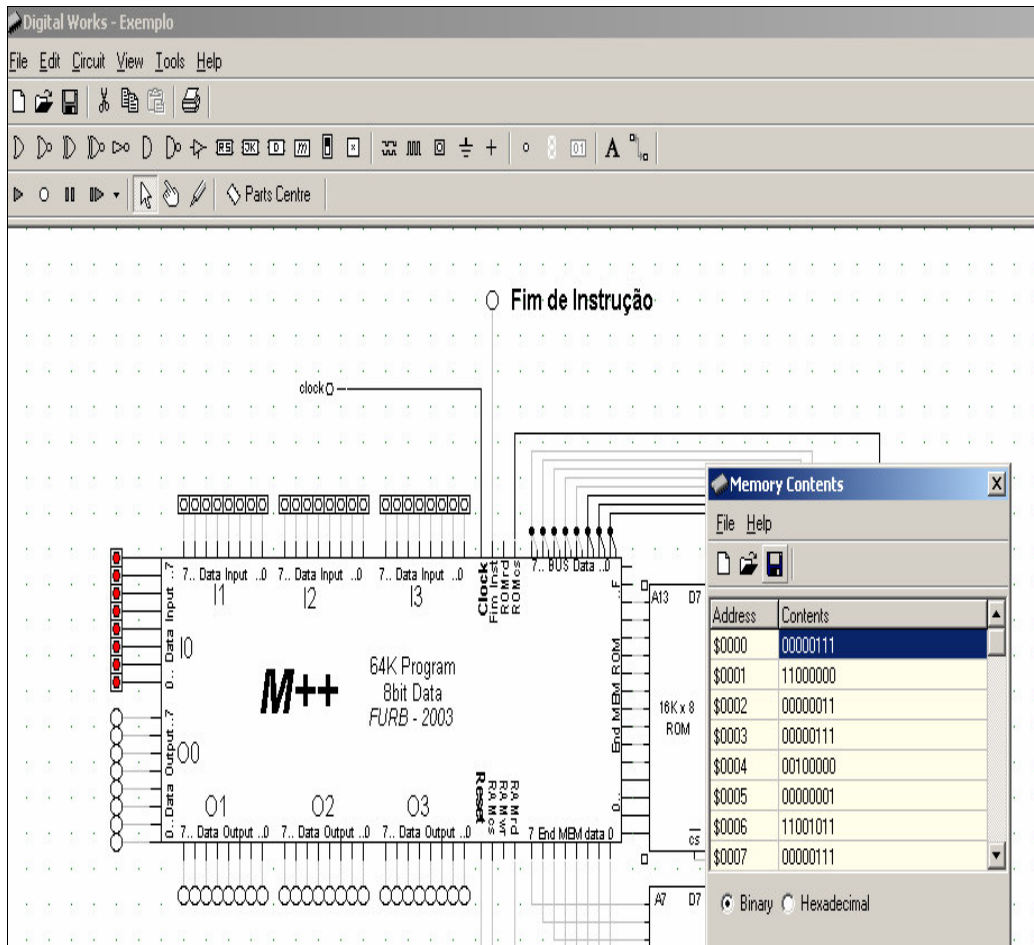





Figura 31 – Carregando um programa em código de máquina na RAM

Pressione a tecla play, click 2 vezes no botão de *reset*, de forma que ele acenda e apague (**Reset** ) , isto resetará a UCP (certifique-se que o ícone  esteja habilitado), acompanhe os sinais elétricos gerados (barramentos). Observe também o dispositivo de saída (E/S) chamado O1, o resultado da execução pode ser observado na figura 32, nela aparecerá o resultado da soma, 55 (hexadecimal) mais 66 (hexadecimal) = BB (hexadecimal), que corresponde a 10111011 em binário .

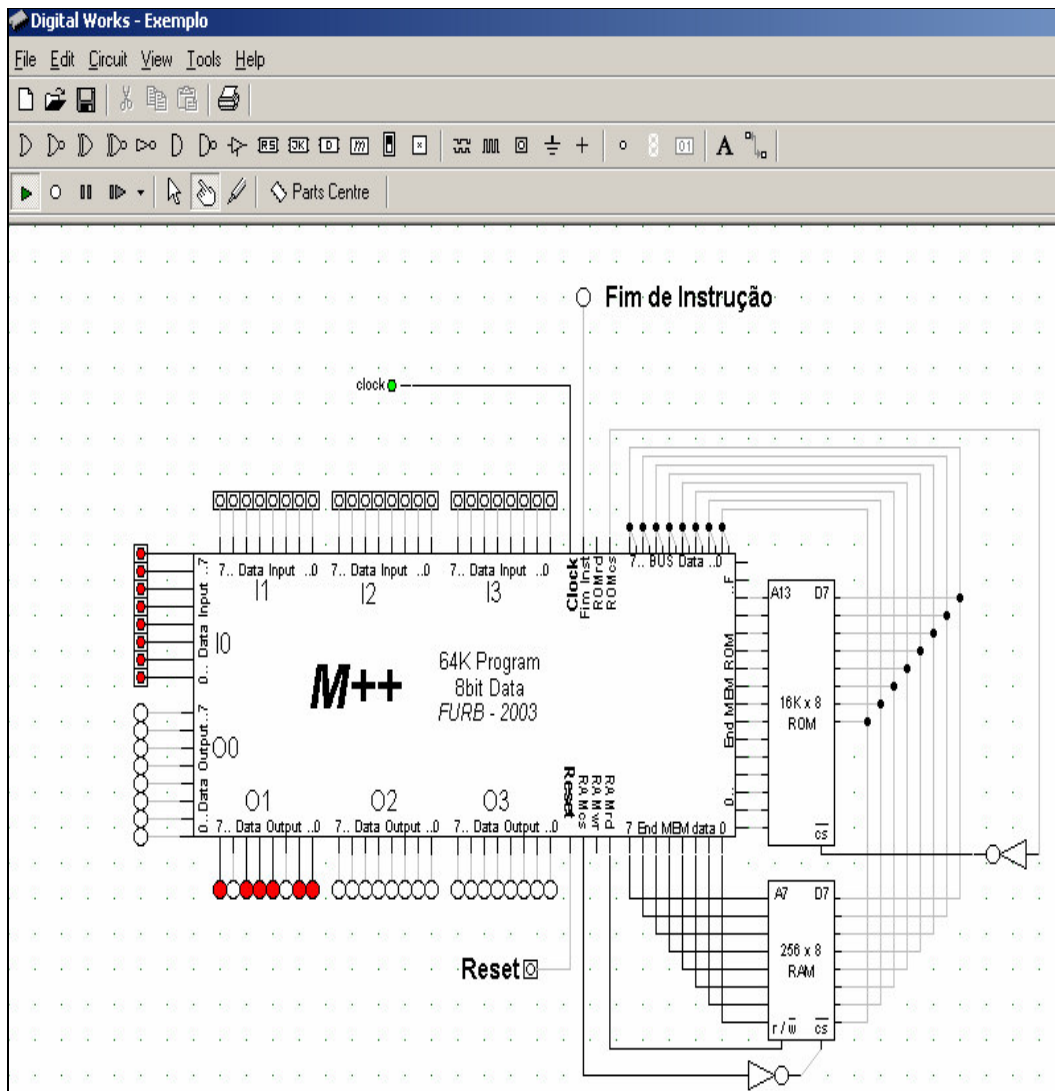


Figura 33 – Execução do programa de soma

A UCP possui um Set de instruções básico que permite resolver uma série de problemas computacionais. Veja a tabela.

ADD *	Operação Aritmética ADD
SUB *	Operação Aritmética SUB
AND *	Operação Lógica E
OR *	Operação Lógica OR
XOR *	Operação Lógica XOR
NOT *	Operação Lógica NOT
MOV *	Movimenta dados
INC *	Incrementa
JMP REND (#0000 - #FFFF)	Desvio incondicional
JMPC REND	Desvio incondicional
JMPZ REND	Desvio incondicional

CALL REND	Chamada de Subrotina
RET	Retorno de subrotina

O *set* de instruções trabalha com uma série de operandos que podem ser manipulados pelas instruções. Veja a tabela.

A, A	Move acumulador para acumulador
A, REG (B, C, D, E)	Move acumulador para B, C, D ou E
A, RAM (#00 - #FF)	Move acumulador para RAM externa
A, OUT (OUT0 - OUT3)	Move acumulador para porta de saída
REG, A	Move registrador para acumulador
RAM, A	Move conteúdo da RAM para acumulador
IN, A (IN0 - IN3)	Le conteúdo da porta de entrada para o acumulador
ROM, A (00 - FF)	⊗
ROM, REG	⊗
ROM, RAM	⊗

Alarme Residencial

O exemplo ilustra a implementação de um alarme residencial que monitora um sensor quando o mesmo é acionado, sinalizando em uma lâmpada no E/S OUT1.. O sensor está conectado no pino 6 (bit 5) da E/S IN0.

APAGADO:	Rotulo para desvio
MOV IN0, A;	Le estado de IN0 e joga no acumulador
AND 32, A;	Faz um AND com acumulador
JMPZ APAGADO;	Se retorna zero, significa que estado de IN0 é zero
PISCA:	Rotulo para o pisca pisca
MOV 01, A;	Joga 1 para acumulador
MOV A, OUT1;	Escreve em OUT1
MOV 00, A;	Joga 1 para acumulador
MOV A, OUT1;	Escreve em OUT1

JMP PISCA;	Desvia para rótulo PISCA
------------	--------------------------

O programa lê o estado da E/S IN0 e isola o pino 5 fazendo uma operação lógica “E” com 32, se e o resultado da operação for Zero, significa que o sensor não está acionado, ficando em *loop*. Quando o resultado for diferente de zero, sai do *loop* e aciona um *led* o qual ficará piscando.

O resultado da execução pode ser visto na seqüência de figuras: figura 34, figura 35 e figura 36.

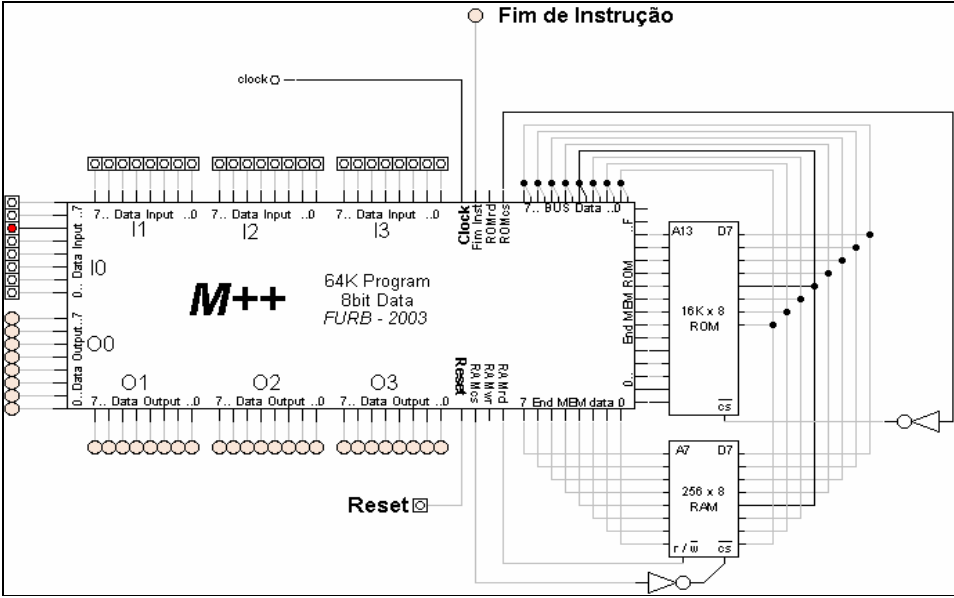


Figura 37 – Simulando acionamento do sensor

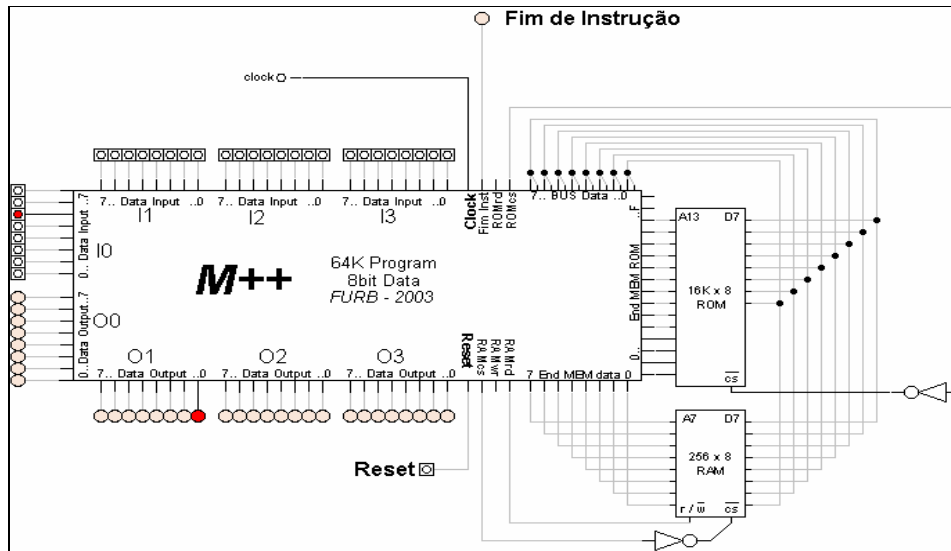


Figura 38 – Led acendendo

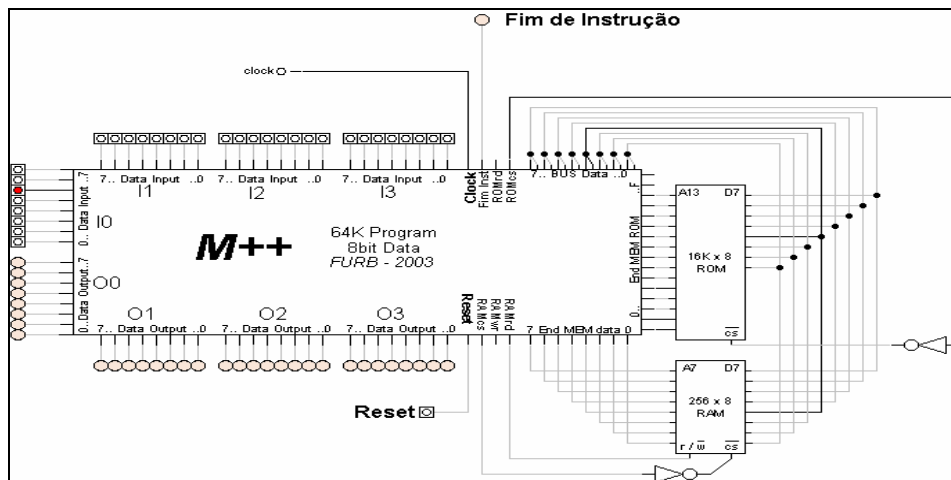



Figura 39 – Led apagando

Execução passo a passo

Para se obter uma execução passo a passo (instrução por instrução), foi implementado um circuito que detecta o fim de execução de uma instrução, paralisando parcialmente o *clock*. Para utilizá-lo, ative “ATIVAR MACROINSTRUÇÃO”  e resete “RESET” (pulso) o circuito, aplique um pulso em “CLOCK” e aguarde “INDICADOR DE FIM DA INSTRUÇÃO” sinalizar, agora navegue pelas macros. Veja figura 40, para prosseguir com a

próxima instrução, aplique novamente um pulso em “CLOCK” e observe os sinais gerados.

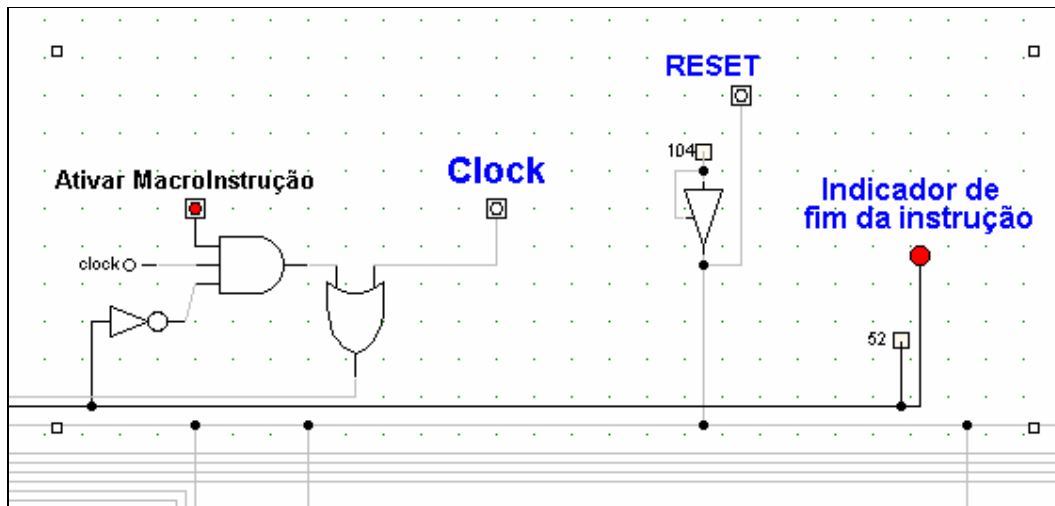


Figura 41 – Ativando macroinstrução

Obtendo a versão DEMOWARE do Digital Works

Digital Works 3.04.39 pode ser encontrado no seguinte site: http://linus.highpoint.edu/~rshore/class/cs341_edp/install.html, o modelo pode ser pego em <http://www.inf.furb.br/~maw/arquitetura/maquina++.dwm>, os exemplos em [EXEMPLOS M++.txt](#) e por fim o montador, em <http://www.inf.furb.br/~maw/arquitetura/AssemblerM++.exe>.

CONCLUSÃO

Para testar o funcionamento da UCP, 12 exemplos foram criados e testados. Alguns *bugs* foram detectados, mas corrigidos na memória de controle. Acreditamos que a UCP não tenha mais BUGs na execução dos programas, gostaríamos que vocês leitores interessados, ajude-nos a achá-los testando os seus próprios programas.