



Universidade de São Paulo - São Carlos, SP

Instituto de Ciências Matemáticas e de Computação

USP – ICMC – SSC

SSC0610 - Organização de Computadores

Professor responsável: *Fernando Santos Osório*

Semestre: 2010/2

Horário: Seg. 10h / Qui. 16h

E-mail: fosorio .at. icmc.usp.br

fosorio .at. gmail.com

Web: <http://www.icmc.usp.br/~fosorio/>

TRABALHO PRÁTICO Nro. 01

Definição de 27/08/2010 (versão 0.1a)

[Descrição Geral]

Este trabalho consiste em implementar um simulador de um microprocessador hipotético, o HSapiens, de acordo com as especificações indicadas logo a seguir. O simulador será “inspirado” no Computador Hipotético Neander, abordado em sala de aula, devendo permitir a execução de programas através da simulação das instruções deste, de sua arquitetura interna, do acesso a memória e aos dispositivos de E/S.

>> Simulador HSapiens <<

Arquitetura do Processador:

- Barramento de Dados: 8 bits
- Barramento de Endereços: 16 bits (memória de até 64KB)
- Registradores Internos de Uso Geral: AC, RX, RY (3 registradores de 8 Bits)
- Registradores: PC (Program Counter – 16 bits), SP (Stack Pointer – 16 Bits), IR (Instruction Register – 8 bits), Flags: N (Bit7), Z (Zero), C (Carry:VaiUm/VemUm)
- Modos de Endereçamento: Imediato, Direto Absoluto, Indireto, Indexado, Registrador (conforme especificado por cada instrução)
- Controle de Entrada e Saída: Instruções dedicadas de E/S

CONJUNTO DE INSTRUÇÕES DO PROCESSADOR:

Load/Store	Bytes	Modo End.	Resultado	Código de Máquina	Flags Afetados
LDA #Valor	2	Imediato	AC <= #Valor	01 #Valor	N,Z
LDA End	3	Direto	AC <= Mem(End)	02 End-Lo End-Hi	N,Z
LDA I,(End)	3	Indireto	AC <= Mem(Mem(End))	03 End-Lo End-Hi	N,Z
LDA End,RX	3	Indexado	AC <= Mem(End+RX)	04 Endereço	N,Z
LDA RX	1	Registrador	AC <= RX	05	N,Z
LDA RY	1	Registrador	AC <= RY	06	N,Z
STA End	3	Direto	Mem(End) <= AC	07 End-Lo End-Hi	N,Z
STA I,(End)	3	Indireto	Mem(Mem(End))<=AC	08 End-Lo End-Hi	N,Z
STA End,RX	3	Indexado	Mem(End+RX) <= AC	09 Endereço	N,Z
STA RX	1	Registrador	RX <= AC	0A	N,Z
STA RY	1	Registrador	RY <= AC	0B	N,Z
LDSP #Valor16	3	Imediato	SP <= #Valor16	0C Valor-Lo Valor-Hi	-
LDX #Valor	2	Imediato	RX <= #Valor	0D #Valor	-
LDY #Valor	2	Imediato	RY <= #Valor	0E #Valor	-
STXY End	3	Direto	Mem(End) <= RX; Mem(End+1) <= RY	0F End-Lo End-Hi	-

ULA-Arit./Log.	Bytes	Modo	End.	Resultado	Código de Máquina	Flags Afetados
ADC	#Valor	2	Imediato	$AC \leq AC + \#Valor + C$	10 #Valor	N,Z,C
ADC	End	3	Direto	$AC \leq AC + Mem(End) + C$	11 End-Lo End-Hi	N,Z,C
SUB	#Valor	2	Imediato	$AC \leq AC - \#Valor$	12 #Valor	N,Z,C
SUB	End	3	Direto	$AC \leq AC - Mem(End)$	13 End-Lo End-Hi	N,Z,C
INC	AC	1	Registrador	$AC \leq AC + 1$	14	N,Z,C
DEC	AC	1	Registrador	$AC \leq AC - 1$	15	N,Z,C
INC	RX	1	Registrador	$RX \leq RX + 1$	16	N,Z,C
DEC	RX	1	Registrador	$RX \leq RX - 1$	17	N,Z,C
AND	#Valor	2	Direto	$AC \leq AC \text{ AND } AC$	18 #Valor	N,Z
OR	#Valor	2	Direto	$AC \leq AC \text{ OR } AC$	19 #Valor	N,Z
NOT		1	Registrador	$AC \leq \text{NOT}(AC)$	1A	N,Z
SLA		1	Registrador	$AC \leq \text{ShiftLeft}(AC)$	1B	N,Z,C
SRA		1	Registrador	$AC \leq \text{ShiftRight}(AC)$	1C	N,Z,C
CPM	#Valor	2	Imediato	Subtract($AC - \#Valor$); AC não é alterado	1D #Valor	N,Z,C

PILHA	Bytes	Modo	End.	Resultado	Código de Máquina	Flags Afetados
PSH	AC	1	Registrador	$Mem(SP) \leq AC$; $SP \leq SP + 1$	20	N,Z
PSH	RX	1	Registrador	$Mem(SP) \leq RX$; $SP \leq SP + 1$	21	-
POP	AC	1	Registrador	$SP \leq SP - 1$; $AC \leq Mem(SP)$	22	N,Z
POP	RX	1	Registrador	$SP \leq SP - 1$; $RX \leq Mem(SP)$	23	-

DESVIO	Bytes	Modo	End.	Resultado	Código de Máquina	Flags Afetados
JMP	End	3	Direto	$PC \leq End$	31 End-Lo End-Hi	-
JPZ	End	3	Direto	If Z=1: $PC \leq End$	32 End-Lo End-Hi	-
JPNZ	End	3	Direto	If Z=0: $PC \leq End$	33 End-Lo End-Hi	-
JPN	End	3	Direto	If N=1: $PC \leq End$	34 End-Lo End-Hi	-
JPC	End	3	Direto	If C=1: $PC \leq End$	35 End-Lo End-Hi	-
JRZ	#Valor	2	Imed./Relativo	If Z=1: $PC \leq PC + \#Valor$	36 #Valor	
JRZ	RX	1	Reg./Relativo	If Z=1: $PC \leq PC + RX$	37	
JSR	End	3	Direto	$Mem(SP) \leq PC - Lo$; $SP \leq SP + 1$; $Mem(SP) \leq PC - Hi$; $SP \leq SP + 1$; $PC \leq End$	38 End-Lo End-Hi	-
					[PC já aponta p/próxima instrução] [Salva endereço de retorno]	
RTS		1	Reg./Direto	$SP \leq SP - 1$; $PC - Hi \leq Mem(SP)$; $SP \leq SP - 1$; $PC - Lo \leq Mem(SP)$	39	-

E/S+Especiais	Bytes	Modo	End.	Resultado	Código de Máquina	Flags Afetados
INP		1	Registrador	$AC \leq \text{Input}()$	40	N,Z
OUT		1	Registrador	Teclado p/Acumulador $Output \leq AC$	41	N,Z
CLR	Flags	1	Registrador	Acumulador p/Tela $N=0$; $C=0$; $Z=0$	42	N,Z,C
SET	Flags	1	Registrador	$N=1$; $C=1$; $Z=1$	43	N,Z,C
CLR	C	1	Registrador	$C=0$	44	C
SET	C	1	Registrador	$C=1$	45	C
HLT		1	-	Para o processador	50 ou 00	-
NOP		1	-	No Operation	FF (Todos códigos inválidos são NOP)	-

INTERFACE com USUÁRIO:

Menu com opções:

- 1- Carregar Programa em Linguagem de Máquina de um Arquivo
- 2- Simula uma instrução (passo-a-passo)
- 3- Simula instruções até encontrar um HLT (executa programa)
- 4- Exibe Estado do Processador (exibe o estado dos registradores)
- 5- Inspecciona Memória (exibe dados da memória)
- 6- Altera Memória (edita dados da memória)
- 7- Salva Memória (grava em disco uma zona da memória)
- 8- Altera Registrador
- 9- Sair do Simulador.

Descrição das Opções:

1. Carrega Arquivo: O usuário informa um nome de um arquivo texto no formato de entrada (ver descrição logo abaixo), onde este arquivo é lido e os dados são carregados para a memória do computador simulado.
2. Simula Instrução: A próxima instrução apontada pelo PC é executada, e logo a seguir são exibidas as seguintes informações na tela: (em hexadecimal)
PC: XXXX - Mem(PC,PC+1,PC+2): YY ZZ WW **SP:** XXXX - Mem(SP-1,SP-2): YY ZZ
RI: XX **AC:** XX **RX:** XX **RY:** YY **Flags:** N=0/1 C=0/1 Z=0/1
MEM: 16 bytes a partir do último endereço inspecionado (opção 4)
INPUT: 16 últimos valores lidos pela instrução input (fila circular)
OUTPUT: 16 últimos valores escritos pela instrução output (fila circular)
3. Simula Execução: Executa o programa até encontrar uma instrução HLT, sendo que só quando terminar a execução no HLT é que vai exibir as informações do mesmo modo que aparecem na opção 2. Sendo assim, a simulação deve executar diversas instruções, porém só atualiza a tela com as informações sobre os registradores e memória ao terminar a execução do programa.
4. Exibe Estado do Processador: Exibe o estado dos registradores do processador, a exemplo do que é feito na opção 2 (com as mesmas informações exibidas nesta outra opção), porém não executa nenhuma instrução nem altera o estado do processador ao exibir estas informações. Apenas consulta o estado dos registradores e mostra na tela da seguinte forma:
PC: XXXX - Mem(PC,PC+1,PC+2): YY ZZ WW **SP:** XXXX - Mem(SP-1,SP-2): YY ZZ
RI: XX **AC:** XX **RX:** XX **RY:** YY **Flags:** N=0/1 C=0/1 Z=0/1
MEM: 16 bytes a partir do último endereço inspecionado (opção 4)
INPUT: 16 últimos valores lidos pela instrução input (fila circular)
OUTPUT: 16 últimos valores escritos pela instrução output (fila circular)
5. Inspecciona Memória: O usuário deve fornecer um endereço inicial, onde deverá a seguir ser apresentado o conteúdo da memória dos próximos 64 bytes de memória a partir deste endereço. Exibir da seguinte forma, por exemplo: (considerando o endereço inicial 0x0100 e 16 bytes apresentados por linha)
0100: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
0110: 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
0120: 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F ! " # \$ % & ' () * + , - . /
0130: 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 0 1 2 3 4 5 6 7 8 9 : ; < = > ?

[Opcionalmente pode exibir o código ASCII dos bytes ao final da linha]
[ou ‘.’ para códigos que não podem ser visualizados]
O endereço inicial é guardado e depois usado no caso da exibição de instruções passo-a-passo (opção 2).

6. Altera Memória: O usuário fornece um endereço de memória e um valor que deseja inserir neste endereço. Ambos os valores são fornecidos em hexadecimal, por exemplo:

Digite o endereço: XXXX

Digite o Valor: YY

Armazenar em XXXX o valor YY, ou seja, Mem(XXXX) <= YY

7. Salva Memória: Grava em um arquivo texto uma zona da memória especificada pelo usuário, no formato do arquivo de saída definido a seguir. Exemplo:

Nome do Arquivo: DumpMem.txt

Digite o endereço inicial: XXXX

Digite o endereço final: YYYY

Grava em disco os dados da memória que estão nesta zona de XXXX a YYYY em um formato similar ao da opção 4 (Inspecciona Memória).

8. Altera Registrador: Pergunta ao usuário qual registrador ele deseja alterar, e depois altera o seu conteúdo, substituindo pelo valor fornecido pelo usuário. Exemplo:

Qual registrador: PC

Qual Registrador: RX

Qual Registrador: FC

Valor: XXXX

Valor: XX

Valor: 1

O usuário poderá selecionar uma das seguintes opções de registradores: PC, SP, AC, RX, RY ou as Flags, nomeadas por FN (Flag Negativo), FC (Flag Carry) e FZ (Flag Zero).

Observação: Se durante a execução de uma instrução for realizado uma instrução INP, o programa deve parar momentaneamente a execução, ler uma tecla do teclado e repassar o código da tecla (código ASCII) para o acumulador, atualizando a fila circular que guarda as últimas 16 entradas. Se for realizada uma instrução OUT, o programa deve parar momentaneamente a execução, exibir na tela o código repassado pelo acumulador (código ASCII), atualizando a fila circular que guarda as últimas 16 saídas.

ARQUIVO de ENTRADA:

Arquivo texto contendo os programas em linguagem de máquina e dados a serem lidos para a memória do computador simulado. O arquivo será composto por um conjunto de blocos de memória seguidos pelos dados correspondentes a serem armazenados nesta área de memória. Formato:

```
XXXX:
YY YY YY YY YY YY ...
YY YY YY YY YY ... YY
#
XXXX:
YY YY YY YY ... YY
#
XXXX:
YY YY YY YY YY YY ... YY
$
```

O arquivo começa com um endereço em hexadecimal que indica onde os dados devem ser armazenados, este endereço é seguido por uma coleção de bytes também representados em hexadecimal e separados por um espaço em branco entre eles. Note que podemos ter diversas linhas de dados com bytes em hexadecimal, até que seja encontrado um caractere especial '#' que indica que um novo endereço será fornecido (seguido de dois pontos e uma série de bytes). Se for encontrado um caractere especial '\$', significa que o arquivo de entrada não possui mais dados (fim do arquivo).

Exemplo prático do formato do arquivo de entrada:

```
0100:
01 02 03 04 05 06 07
#
12F0:
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1F
20 21 22 23
10A0:
FF FF FF FF
$
```

ARQUIVO de SAÍDA:

Arquivo contendo um dump (imagem) da memória do computador. Considerando o endereço inicial e final definido pelo usuário, escrever no arquivo texto o conteúdo da memória em formato similar ao da opção 2 (que exibe na tela). Por exemplo:

Dump de Memória de 0x0100 a 0x013F (poderia abranger toda memória de 0x0000 a 0xFFFF).

```
0100: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F . . . . .
0110: 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F . . . . .
0120: 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F ! " # $ % & ' ( ) * + , - . /
0130: 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
```

[Opcionalmente pode exibir o código ASCII dos bytes ao final da linha]
[ou '.' para códigos que não podem ser visualizados]

ENTREGA DO TRABALHO:

* Envie em um 1º. e-mail anexados os arquivos fonte do projeto de seu trabalho ao prof. Osório (incluir os fontes e a documentação apenas, sem o executável!)

E-MAIL TO: **fosorio@gmail.com** (Enviar o original para este email)
EMAIL CC: **work2usp@yahoo.com** (Enviar com cópia para este email)
SUBJECT: **[SSC0610] TP01-Fonte <nome_aluno>** (Assunto do email)

* Envie em um 2º. e-mail com os arquivos executáveis do projeto de seu trabalho ao prof. Osório (O executável deve ser anexado em formato .RAR, ou em um ZIP com o EXE renomeado para .EXX !)

E-MAIL TO: **fosorio@gmail.com** (Enviar o original para este email)
EMAIL CC: **work2usp@yahoo.com** (Enviar com cópia para este email)
SUBJECT: **[SSC0610] TP01-Exec <nome_aluno>** (Assunto do email)

Cuidado: Se você for enviar qualquer tipo de executável (.exe, .com, .bat, arquivo de scripts), o arquivo terá obrigatoriamente que estar compactado em formato **.rar** ou **.bz2** (OUTROS FORMATOS NÃO SERÃO ACEITOS, POIS SÃO RECUSADOS PELO SERVIDOR DE E-MAIL). O gmail e yahoo não aceitam anexos com executáveis! (tem que “disfarçar”)

* Escreva no corpo das mensagens de e-mail:

NOME: <seu nome> + <Nro. USP>

NOME: <nome dos demais componentes do grupo> + <nro. USP dos demais membros do grupo>

INFORMAÇÕES SOBRE O PROJETO:

<informações que julgar necessárias para a avaliação e teste do seu projeto>

(por exemplo, como executar o programa, como testar com exemplos de uso)

* Entregar até a data indicada no Site da Disciplina / Wiki ICMC

<http://www.icmc.usp.br/~fosorio/> (Trabalhos Práticos)

===== THAT'S ALL FOLKS !!! =====