Pontifícia Universidade Católica de Minas Gerais
Instituto de Ciências Exatas e Informática – ICEI
Ciência da Computação – Arquitetura de Computadores I

ARQ I - ATIVIDADE PRÁTICA

PROJETO DE AMPLIAÇÃO E TESTES DE ARQUITETURA DE CONJUNTO DE INSTRUÇÕES (ISA)

Os arquivos solicitados nas atividades relacionadas abaixo
deverão ser entregues no formato texto (TXT e HTML, **NÃO USAR** .doc ou .docx),
devidamente identificados com nome e matrícula.

INSTRUÇÕES:

1.) Instalar o simulador CPUSim3.9.0.X.zip constante no pacote de programas da disciplina.

2.) Carregar e abrir o exemplo Wombat1.a na pasta *SampleAssignments* (usar Cpusim.bat).

3.) Modificar o conjunto de instruções da linguagem para uma máquina do tipo acumulador

Com a seguinte especificação de registradores:

Registradores	Bits	Descrição
PC	**12**	*Program Counter* = apontador de instrução
IR	8+8	*Instruction Register* = registrador de instrução
AC	16	*ACcumulator* = acumulador
MAR	**12**	*Memory Address Register* = endereço da memória
MDR	16	*Memory Address Register* = dado da memória
STATUS	8	registrador para códigos de condição (*flags*)

Sugestões:

O PC e o MAR deverão ser limitados ao tamanho da memória RAM (128 posições).

No caso do registrador de instrução serão usados 8 bits para o código de operação (*opcode*) e os outros 8 bits serão usados para endereços de operandos ou de instruções na memória, quando necessários.

Com o seguinte conjunto básico de instruções

Mnemônico	Opcode	Formato	Descrição	Algoritmo
NOP	0	8 + _	*No OPeration*	// não fazer nada
LDA	1	8 + 8	*LoaD Accumulator*	AC = MEM [MAR]
STO	2	8 + 8	*STOre accumulator*	MEM [MAR] = AC
(livre)	3			
ADD	4	8 + 8	*ADD value to accumulator*	AC = AC + MEM [MAR]
SUB	5	8 + 8	*SUBtract value from accumulator*	AC = AC - MEM [MAR]
XOR	6	8 + 8	*XOR value with accumulator*	AC = AC ^ MEM [MAR]
AND	7	8 + 8	*AND value with accumulator*	AC = AC & MEM [MAR]
IN	8	8 + _	*INput value to accumulator*	AC = input
OUT	9	8 + _	*OUTput value in accumulator*	output = AC
(livre)	A			
NOT	B	8 + 8	*NOT value in accumulator*	AC = ~AC
BRA	C	8 + 8	*BRAnch Unconditionally*	PC = address
BRZ	D	8 + 8	*BRAnch if Zero*	PC = address, (if AC == 0)
BRP	E	8 + 8	*BRAnch if Positive*	PC = address, (if AC > 0)
HLT	F	8 + _	*HaLT*	// parar

Notas:

a.) Os formatos terão o seguinte significado:

8 + _ = 8 bits para instrução no byte superior e desprezar qualquer valor no byte inferior

8 + 8 = 8 bits para instrução no byte superior e endereço de operando no byte inferior

b.) As operações lógicas deverão ser implementadas por microinstruções equivalentes (Logical).

c.) Implementar duas outras instruções a sua escolha nos códigos livres, que não sejam lógicas ou desvios (*branches*).

4.) Desenvolver programas para

a.) mostrar a tabela-verdade da instrução NAND (**que não deverá ser implementada**);

b.) calcular o quadrado de certo valor (n<16) mediante soma de ímpares;

c.) testar se um valor inteiro **não** é primo;

d.) calcular e mostrar primeiros números **pares** da série de Fibonacci menores que 100;

e.) calcular o Máximo Divisor Comum (M.D.C) entre dois números, usando apenas subtrações.

EXTRA

Desenvolver opcionalmente programas para

1.) ordenar um arranjo em ordem decrescente pelo método da Bolha;

2.) procurar por valor em arranjo usando pesquisa sequencial rápida.