PUCRS - Escola Politécnica Disciplina: Sistemas Operacionais - 2020/1 - Trabalho Prático Prof. Fernando Luís Dotti

1. Definição da Máquina Virtual (MV)

Nossa máquina virtual (MV) tem CPU e Memória.

1.1 Memória

Considere a memória como um array contíguo de posições de memória. Cada posição de memória são 4 bytes. A memória tem 1024 posições.

tamMemoria = 1024

array mem[tamMemoria] of posicaoDeMemoria // adotamos 32 bits

1.2 CPU - versão 3

O processador possui os seguintes registradores:

- um contador de instruções (PC)
- oito registradores, 0 a 7

O conjunto de instruções é apresentado na tabela a seguir, adaptado de [1].

					16-bit Instruction Format																										_	_			
					16-bit Instruction Format																														
						OPCODE					PARAMETERS									PARAMETERS															
No.	Mne moni c	Descripti on	Syntax	Micro- operation	15	14	13	12	11	10	9	8	7	6	5	4 3	3 2	2 1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
J - Type Instructions																																			
1	JMP	Direct Jump	JMP k	PC ← k	0	0	0	0	0	-	-	-	-		-		. -	. -	-	k	k	k	k	k	k	k	k	k	k	k	k	k	k	k	k
2	JMPI			PC ← Rs	0	0	0	0	1	Rs	Rs I	Rs	-	-	-	.	. -	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
3	JMPI G			if Rc > 0 then PC ← Rs Else PC ← PC +1	0	0	0	1	0	Rs	Rs I	Rs I	Rc F	Rc F	Rc -	. .	. -	. -	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
4	JMPI L			if Rc < 0 then PC ← Rs Else PC ← PC +1	0	0	0	1	1	Rs	Rs I	Rs I	Rc F	Rc F	Rc .	.	. -	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
5	JMPI E			if Rc = 0 then PC ← Rs Else PC ← PC +1	0	0	1	0	0	Rs	Rs I	Rs I	Rc F	Rc F	Rc -			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	I - Type Instructions																																		
6	ADDI	Immedia te addition	ADDI Rd, k	Rd ← Rd + k	0	1	0	0	0	Rd	Rd I	Rd	-		.				-	k	k	k	k	k	k	k	k	k	k	k	k	k	k	k	k
7	SUBI	Immedia te subtracti on	SUBI Rd, k	Rd ← Rd – k	0	1	0	0	1	Rd	Rd I	Rd	-		-			-	-	k	k	k	k	k	k	k	k	k	k	k	k	k	k	k	k
8	ANDI	Immedia te AND	ANDI Rd, k	Rd ←Rd AND k	0	1	0	1	0	Rd	Rd I	Rd	-	-	-	.		-	-	k	k	k	k	k	k	k	k	k	k	k	k	k	k	k	k
9	ORI	Immedia te OR	ORI Rd, k	Rd ←Rd OR k	0	1	0	1	1	Rd	Rd I	Rd	-	-	-		-	-	-	k	k	k	k	k	k	k	k	k	k	k	k	k	k	k	k
10	LDI	Load immedia te	LDI Rd, k	Rd ← k	0	1	1	0	0	Rd	Rd I	Rd	-		-	.		-	-	k	k	k	k	k	k	k	k	k	k	k	k	k	k	k	k

11	LDD	Load direct from data memory	LDD Rd,[A]	Rd ← [A]	0	1	1	0	1	Rd F	Rd F	Rd	- -		-	-	-	-		A A	А	А	А	А	Α	Α	А	А	Α	А	Α	А	А	А
12	STD	Store direct to data memory	STD [A],Rs	[A] ← Rs	0	1	1	1	0 1	Rs F	Rs F	is.			-	-	-		. <i>p</i>	A A	А	А	Α	Α	Α	Α	Α	Α	Α	Α	Α	Α	Α	А
								ı	R2 -	- Тур	e Ins	struc	ction	s																				
13	ADD	Addition	ADD Rd, Rs	Rd ← Rd + Rs	1	0	0	0	0 F	Rd F	Rd F	Rd F	Rs R	s Rs	-	-	-		.	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
14	SUB	Subtract ion	SUB Rd, Rs	Rd ← Rd - Rs	1	0	0	0	1 [Rd F	Rd F	Rd F	Rs R	s Rs	-	-	-	-	. -	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
NO VA	MUL T	multiplic ation	MULT Rd, Rs	Rd ← Rd * Rs	1	1	1	0	0 F	Rd F	Rd F	Rd F	Rs R	s Rs	-	-	-	-	. -	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	AND	Logic AND	AND Rd, Rs	Rd ← Rd AND Rs	1	0	0	1	0 1	Rd F	Rd F	Rd F	Rs R	s Rs	-	-	-	-	. -	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
16	OR	Logic OR	OR Rd, Rs	Rd ← Rd OR Rs	1	0	0	1	1 F	Rd F	Rd F	Rd F	Rs R	s Rs	-	-	-		. -	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
17	LDX	Indirect load from memory	LDX Rd,[Rs]	Rd ← [Rs]	1	0	1	0	0 F	Rd F	Rd F	Rd F	Rs R	s Rs	-	-	-		- -	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
18	STX	Indirect storage to memory	STX [Rd],Rs	[Rd] ←Rs	1	0	1	0	1 F	Rd F	Rd F	Rd F	Rs R	s Rs	; -	-	-			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	R1 – Type Instructions																	1																
19	NOT	Logic NOT	NOT Rd	Rd←NOT Rd	1	1	0	0	0 F	Rd F	Rd F	₹d		. -	-	-	-	-	. -	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
20	SHL	Shift register left	SHL Rd	$Rd(n+1) \leftarrow Rd(n),$ $Rd(0) \leftarrow 0$	1	1	0	0	1	Rd F	Rd F	Rd	- -	-	-	-	-	-	. .	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
21	SHR	Shift register right	SHR Rd	$Rd(n) \leftarrow Rd(n+1),$ $Rd(7) \leftarrow 0$	1	1	0	1	0 F	Rd F	Rd F	₹d			-	-	-	-		-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
22	SWA P	Swap nibbles	SWAP Rd	Rd7←Rd3, Rd6←Rd2, Rd5←Rd1, Rd4←Rd0	1	1	0	1	1 F	Rd F	Rd F	₹d			-	-			. -	-	-	-	-	-		-	-	-	-	-	-	-	-	-
23	STO P				1	1	1	1	1	-		-	-		-	-	-	-	. -	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

1.3 Programas

Construa os seguintes programas para a nossa MV, no mínimo os que seguem:

- a) P1: o programa escreve em posições sabidas de memórias os 10 números da sequência de fibonacci. Ou seja, ao final do programa a memória tem estes 10 números em posições convencionadas no programa.
- b) P2: o programa le um valor de uma determinada posição (carregada no inicio), se o número for menor que zero coloca -1 no início da posição de memória para saída;

se for maior que zero este é o número de valores

da sequencia de fibonacci a serem escritos em sequencia a partir de uma posição de

memoria:

c) P3: dado um inteiro em alguma posição de memória,
 se for negativo armazena -1 na saída;
 se for positivo responde o fatorial do número na saída.

d) P4: para um N definido (5 por exemplo)

o programa ordena um vetor de N números em alguma posição de memória; ordena usando bubble sort

loop ate que nao swap nada passando pelos N valores faz swap de vizinhos se da esquerda maior que da direita

1.4 Shell

Construa um shell, ou seja, um terminal interativo que:

- · oferece um prompt ao usuário
- o usuário pode solicitar a execução de um programa
- a partir disso
 - · o programa é carregado
 - executado
 - · ao acabar retorna controle para o shell
- para armazenar o programa, pode-se adotar
 - · um arquivo por programa, ou mesmo
 - que existe uma estrutura em memória com os programas que são então carregados para a memória da VM.

2. PROBLEMA

Implemente: a MV descrita acima; o Shell; e rotinas necessárias.

Defina como acontece a carga do programa em memória e o início da execução.

Execute os programas P1 a P4 na MV.

Instrumente para ver o andamento do processamento na MV.

Referências:

[1] Design of a General Purpose 8-bit RISC Processor for Computer Architecture Learning. Antonio Hernández Zavala, Oscar Camacho Nieto, Jorge A. Huerta Ruelas, Arodí R. Carvallo Domínguez. Computación y Sistemas, Vol. 19, No. 2, 2015, pp. 371–385 ISSN 1405-5546 doi: 10.13053/CyS-19-2-1941