Prática de Assembly

Arquitetura de Organização de Computadores

https://sites.google.com/ic.ufal.br/ecom025

Profº Erick de A. Barboza

UNIVERSIDADE FEDERAL
DE ALAGOAS

Previously...

Código C

```
if (i == h)
  i  = g + h;
else
  i = g - h;
```

g em $s1, h em $s2, i em $s3

Código Assembly MIPS

```
bne $s3, $s2, Else
add $s3, $s1, $s2
j Exit
Else: sub $s3,$s1,$s2
Exit:...
```

Instrução	Descrição
nop	No operation
lw reg, end(reg_base)	reg. = mem (reg_base+end)
sw reg, end(reg_base)	Mem(reg_base+end) = reg
add regi, regj,regk	Regi. <- Regj. + Regk
sub regi, regj, regk	Regi. <- Regj. – Regk
and regi, regj,regk	Regi. <- Regj. and Regk
xor regi, regj, regk	Regi = regj xor regk
srl regd, regs, n	Desloca regs para direita n vezes sem preservar sinal, armazena valor deslocado em regd .
sra regd, regs, n	Desloca regs para dir. n vezes preservando o sinal, armazena valor deslocado em regd.
sll regd, regs, n	Desloca regs para esquerda n vezes, armazena valor deslocado em regd.
ror regd, regs, n	Rotaciona regs para direita n vezes, armazena valor deslocado em regd.
rol regd, regs, n	Rotaciona regs para esquerda n vezes, armazena valor deslocado em regd.
beq regi, regj, desl	PC = PC + desl*4 se regi = regj
bne regi, regj, desl	PC = PC + desl *4 se regi <> regj
slt regi, regj, regk	Regi =1 se regj < regk senão regi=0
j end	Desvio para end
jr reg	Pc = reg
jal end	Reg31 = pc, pc = endereço
break	Para a execução do programa

Exercício 1

Codifique um programa correspondente ao seguinte pseudo-código:

```
int a = 2;
int b = 1;
int m = 0;
m = a;
if ( b < m )
    m = b;
else
    m = 0;
```

Exercício 2

Codifique um programa correspondente ao seguinte pseudo-código:

```
int a = ...;
int b = ...;
int x = 0;

if ( a >= 0 && a < b )
    x = 1;
else if( a < 0 && a > b )
    x = 2;
else
    x = 3;
```

Manipulação de string no MIPS

UNIVERSIDADE FEDERAL DE ALAGOAS

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Manipulação de string no MIPS

- Instruções para leitura/escrita de bytes

```
lb $t0, 0($sp)     #Ler byte da origem
sb $t0, 0($gp)     #Escreve bytes no destino
```

Manipulação de string no MIPS

UNIVERSIDADE FEDERAL DE ALAGOAS

Código em C

```
void strcpy(char x[], char y[])
{
     int i;
     i=0;
     while((x[i]=y[i]) != '\0')
     i+=1;
}
```

```
$a0 = x[]
$a1 = y[]
```

Código em Assembly

```
strcpy:
     addi $sp,$sp,-4    # adjust stack for 1 more item
     sw $s0, 0($sp)          # save $s0
     add $s0,$zero,$zero     # i = 0 + 0
L1:  add $t1,$s0,$a1    # address of y[i] in $t1
     lbu $t2, 0($t1)    # $t2 = y[i]
     add $t3,$s0,$a0    # address of x[i] in $t3
     sb $t2, 0($t3)     # x[i] = y[i]
     beq $t2,$zero,L2   # if y[i] == 0, go to L2
     addi $s0, $s0,1    # i = i + 1
     j L1               # go to L1
L2: lw $s0, 0($sp)      # y[i] == 0: end of string.
                        # Restore old $s0
     addi $sp,$sp,4     # pop 1 word off stack
     jr $ra             # return
```

Exercício 3

Faça um programa em linguagem de montagem Mips que receba como entrada uma string com *n* caracteres e gere como saída uma nova string com a inversão da ordem dos caracteres. Essa nova string também terá a troca das letras maiúsculas por minúsculas e vice-versa. Por exemplo: se a entrada for HArdwArE a saída deverá ser eRaWDRah. A entrada deve ser lida da memória e a saída deve ser escrita na memória. Caso a string possua algum caractere que não seja letra o valor 1 deverá ser armazenado no registrador v1 e o programa deverá ser encerrado. Lembramos que o fim da string é dado pelo caractere nulo e que para manipular string nessa questão recomendamos que sejam usadas variáveis do tipo ASCII.

Atenção: os caracteres deverão ser armazenados em sequência na memória.

To be continued...

$$\text{Tempo de CPU} = \frac{\text{Instruções}}{\text{Programa}} \times \frac{\text{Ciclos de Clock}}{\text{Instrução}} \times \frac{\text{Segundos}}{\text{Ciclo de Clock}}$$

Dúvidas?