Arquitetura e Organização de Computadores

Suporte a chamadas de procedimentos e funções em hardware

Prof. André D'Amato

andredamato@utfpr.edu.br

Suporte Mips a chamadas de procedimentos

- Execução de procedimento
  - Colocar parâmetros em um lugar onde o procedimento possa acessá-los;
  - Transferir o controle para o procedimento;
  - Adquirir os recursos de armazenamento necessários para o procedimento;
  - Realizar a tarefa desejada;
  - Colocar o valor de retorno em um local onde o programa que o chamou possa acessá-lo;
  - Retornar o controle para o ponto de origem, pois um procedimento pode ser chamado de vários pontos em um programa.

Suporte Mips a chamadas de procedimentos

- MIPS utiliza a seguinte convenção na alocação de seus 32 registradores:
  - $a0-$a3 : quatro registradores de argumento, para passar parâmetros;
  - $v0-$v1 : dois registradores de valor, para valores de retorno;
  - $ra : um registrador de endereço de retorno, para retornar ao ponto de origem.
- Jal Endereço_de_retorno
- Jr $ra

Suporte Mips a chamadas de procedimentos

- A estrutura de dados ideal para armazenar os “spilled registers” é uma pilha;
- O stack pointer é ajustado a cada registrador salvo ou restaurado.
  - Operações **push** e **pop**;
- O software MIPS reserva o registrador 29 para o stack pointer, dando-lhe o nome óbvio $sp.

Compilando um procedimento em C

```
Int folha(int g, int h, int I, int j){

        int f;
        f = (g + h) - (I + j);
        return f;

}
```

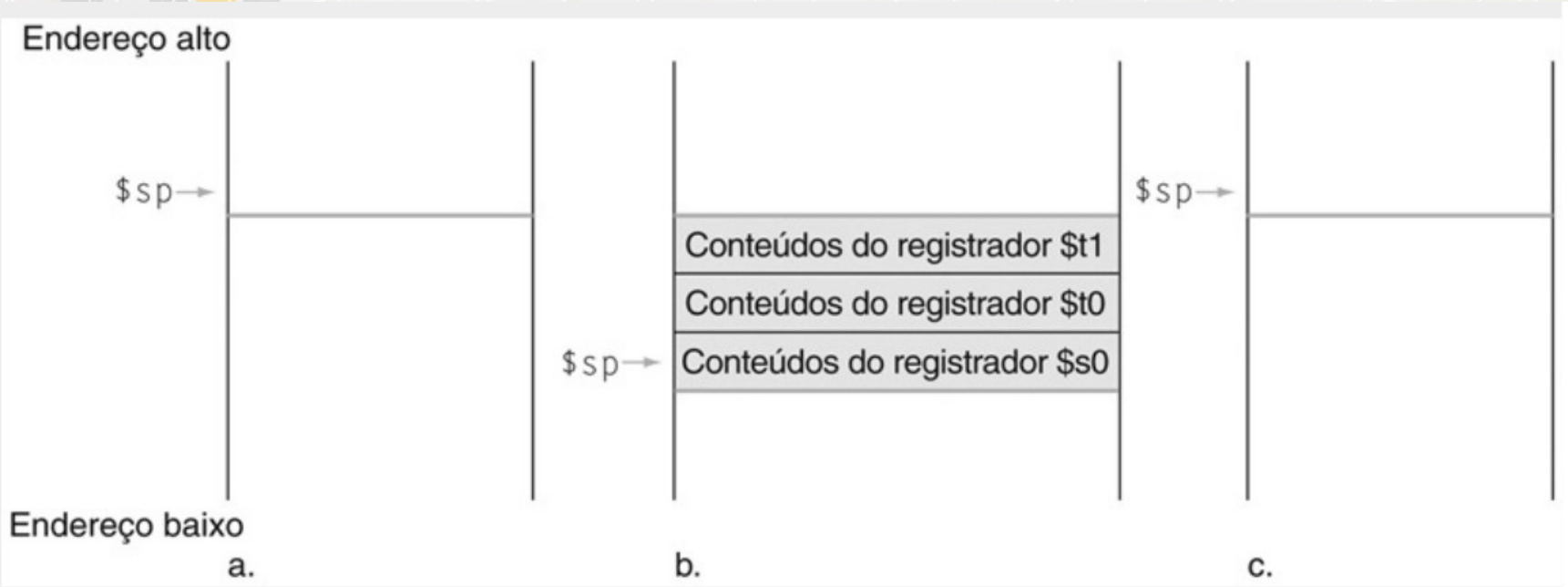
- Como ficaria o código assembly?

Compilando um procedimento em C

- Salvando os registradores para utilização

```
addi  $sp, $sp, -12
sw  $t1, 8($sp)
sw  $t0, 4($sp)
sw  $s0, 0($sp)
```

Compilando um procedimento em C



- Salvando os registradores para utilização

```
addi  $sp, $sp, -12
sw  $t1, 8($sp)
sw  $t0, 4($sp)
sw  $s0, 0($sp)
```

Compilando um procedimento em C

- Fazendo os cálculos

```
addi  $sp, $sp, -12
sw  $t1, 8($sp)
sw  $t0, 4($sp)
sw  $s0, 0($sp)
```

```
add $t0, $a0, $a1   → t0 = g + h
add $t1, $a2, $a3   → t1 = I + j
sub $s0, $t0, $t1   → f = (g+h)-(i+j)
```

Compilando um procedimento em C

- Retornando o valor de f

```
addi  $sp, $sp, -12
sw  $t1, 8($sp)
sw  $t0, 4($sp)
sw  $s0, 0($sp)

add $t0, $a0, $a1
add $t1, $a2, $a3
sub $s0, $t0, $t1

add $v0, $s0, $zero
```

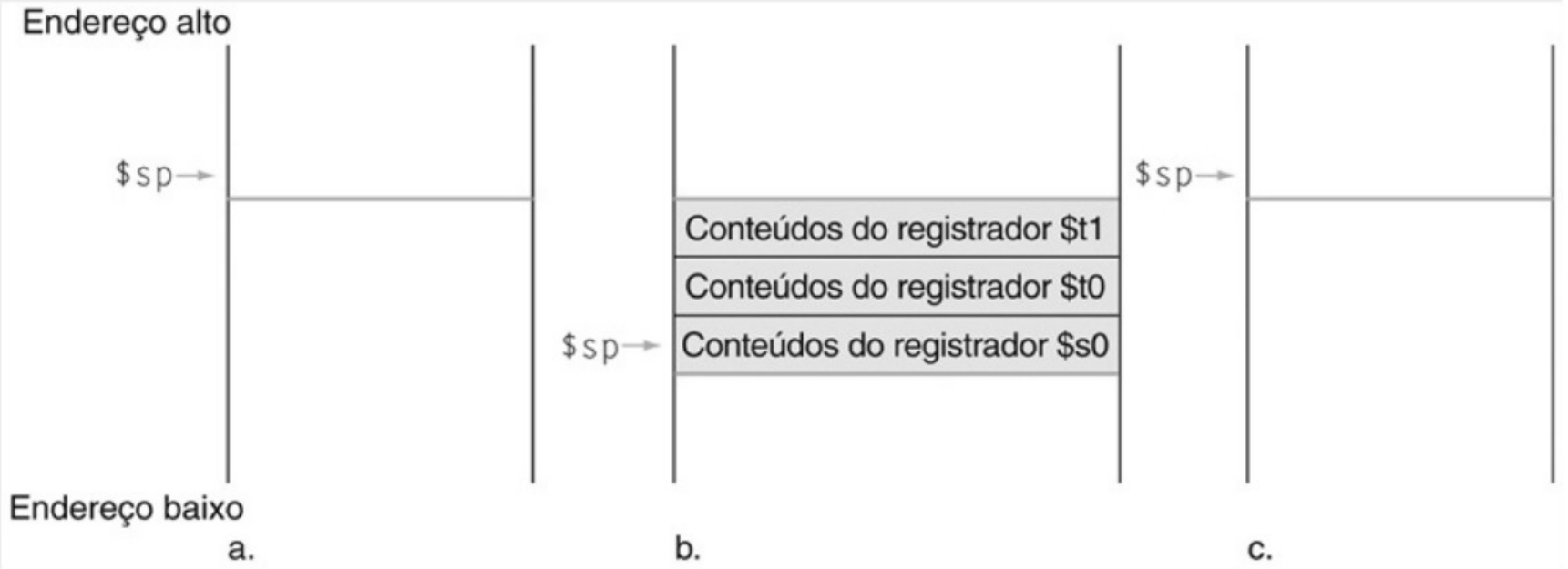
Compilando um procedimento em C

- Recuperando os valores dos registradores

```
add $v0, $s0, $zero

lw  $s0, 0($sp)
lw  $t0, 4($sp)
lw  $t1, 8($sp)
addi  $sp, $sp, 12
```

Compilando um procedimento em C



- Recuperando os valores dos registradores (figura c)

Compilando um procedimento em C

- Retorna para rotina que chamou

```
add $v0, $s0, $zero

lw  $s0, 0($sp)
lw  $t0, 4($sp)
lw  $t1, 8($sp)
addi  $sp, $sp, 12

jr $ra
```

Procedimentos anihados

- Os procedimentos que não chamam outros são denominados procedimentos folha:

  - Chamador: empilhar todos os outros registradores que precisam ser preservados, assim como fizemos com os registradores salvos;

  - O “callee” empilha o registrador do endereço de retorno $ra e quaisquer registradores salvos ($s0 – $s7) usados por ele.

Procedimentos anihados

- Os procedimentos que não chamam outros são denominados procedimentos folha:

  - O “stack” pointer $sp é ajustado para levar em consideração a quantidade de registradores colocados na pilha. No retorno, os registradores são restaurados da memória e o stack pointer é reajustado.

Instruções de Controle

```
int fact(int n){

        if (n < 1) return 1;
                else return (n * fact(n – 1));

}
```

- Como ficaria em assembly?

Strings e caracteres

Valor ASCII	Sinal	Valor ASCII	Sinal	Valor ASCII	Sinal	Valor ASCII	Sinal	Valor ASCII	Sinal	Valor ASCII	Sinal
32	Espaç	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	DEL

Representação dos caracteres em código

Strings

- “Cal” é representada em C pelos 4 bytes a seguir, em forma de números decimais: 67, 97, 108, 0.

```
lb $t0, 0($sp)      # Lê byte de origem
sb $t0, 0($gp)      # Escreve byte de origem
```

Strings

```
void strcpy(char x[], char y[]){

        int i;
        i = 0;
        while( (x[i] = y[i]) != '\0')
            i++;

}
```

Como ficaria em assembly?

Referências

- PATTERSON, David A.; HENNESSY, John L. Organização e projeto de computadores: a interface hardware/software, 5. ed. Rio de Janeiro: Elsevier, 2017. ISBN 9788535287936.

- STALLINGS, William.; Arquitetura e Organização de Computadores. 10. ed. São Paulo:Pearson Education do Brasil, 2017. ISBN 9788543020532.

- TANENBAUM, A. S.; AUSTIN, T. Organização Estruturada de Computadores. 6. ed. SãoPaulo: Pearson Education do Brasil, 2013. ISBN 9788581435398.