

# ROTINAS E PILHA

chamada e retorno de rotinas

- $pc \rightarrow a_0, L \rightarrow$  quando em  $a_0$  o endereço da próxima instrução e pula para o retículo  $L$   
 $\rightarrow PC+4 = ra$  (return address)
- $ret \rightarrow$  salto indireto para a função  $ra$
- problema  $\rightarrow$  quando uma rotina chama uma subrotina, perdemos o endereço de  $ra$ !
- solução  $\rightarrow$  salvar o endereço de retorno, antes de invocar uma rotina!

$\rightarrow$  SALVAR NA PILHA DO PROGRAMA!  $\rightarrow$

pilha do programa

- rotina ativa  $\rightarrow$  rotina que não retornou  $\rightarrow$  necessidade de armazenar seus dados
- antes de retornar, o código da própria rotina desloca o espaço alocado na pilha
- cresce para baixo  $\rightarrow$  começa da posição de memória maior e vai diminuindo  
 $\rightarrow$  última posição preenchida na pilha
- SP  $\rightarrow$  Stack Pointer  $\rightarrow$  aponta para o topo da pilha
- empilha o valor de  $a_0 \rightarrow$   

addi $sp, sp, -4$	$\rightarrow$ desloca, e reapto p. que cresce para baixo
sw $a_0, 0(sp)$	$\rightarrow$ aloca o valor
- desempilha o valor de  $a_0 \rightarrow$   
em  $a_1$   

lw $a_1, 0(sp)$	$\rightarrow$ lê o valor
addi $sp, sp, 4$	$\rightarrow$ desloca, $sp$ volta a apontar para o topo anterior
- empilhando vários valores  $\rightarrow$   

addi $sp, sp, -12$
sw $a_0, 8(sp)$
sw $a_1, 4(sp)$
sw $a_2, 0(sp)$
- desempilha vários valores  $\rightarrow$   

lw ...
addi $sp, sp, 1 \cdot n$

 $\rightarrow$  quantidade de valores alocados!
- ATENÇÃO!  $\rightarrow$  último empilhado deve ser o 1º desempilhado!

salvando o endereço de retorno

- "tudo que empilha, desempilha no final"  $\rightarrow$

função:	
addi $sp, sp, -4$	$\rightarrow$ empilha $\rightarrow$ endereço de retorno
sw $ra, 0(sp)$	$\rightarrow$ adiciona $ra$ na pilha
chamada função	$\rightarrow$ chama outra função
lw $ra, 0(sp)$	$\rightarrow$ recupera $ra$ , volta o endereço
addi $sp, sp, 4$	$\rightarrow$ desloca / desempilha

# ABI E A POLÍTICA DE USO DOS REGISTRADORES

## convenção de chamada

- define onde os parâmetros de uma função devem ser passados

- permite a uso global de funções!

## retorno de valores das funções

- até 32 bits  $\rightarrow a_0$

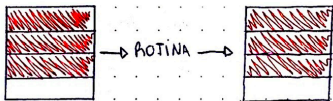
$\uparrow$  parte menos significativa  
 $\uparrow$  parte mais significativa

- entre 32 e 64 bits  $\rightarrow a_0 \pm a_1$

- parâmetros  $\rightarrow a_0, a_1, \dots, a_7$ , sendo que o resto vai pela pilha!

$\uparrow$  empilha 1º o último!

- antes da chamada, os parâmetros são empilhados e só depois de executar a rotina, são desempilhados!



- eles  $\rightarrow$  empilha 1º o último, pois minha função deve ler os valores na ordem correta

## passagem de parâmetros por valor

- li  $a_0$ , valor

## passagem de parâmetros por referência

- antes da função  $\rightarrow$  li  $a_0$ , endereço do valor

- dentro da função  $\rightarrow$  li  $x_1$ ,  $0(a_0)$   
 $\vdots$   
ou  $x_i$ ,  $0(a_p)$

## política de uso de registradores

- callee-saved  $\rightarrow$  outros registradores  $\rightarrow$  registradores salvos pela rotina chamadora

- callee-saved  $\rightarrow r_0$  a  $r_{11}$   $\rightarrow$  registradores salvos pela rotina salva

quadro de pilha  $\rightarrow$  rotinas ativas não adicionando quadros para suprir suas necessidades

- frame pointer  $\rightarrow$  aponta para o início do quadro atual  $\rightarrow$

 $\text{addi } r_p, r_p, -8$   
 $\text{addi } r_p, r_p, 8$