

Elementos de Sistemas

Aula 18 – Operações em Pilhas

"Qualquer tecnologia suficientemente avançada é indistinguível de magia."

"Any sufficiently advanced technology is indistinguishable from magic."

Arthur C. Clarke (1917-2008), Escritor Britânico



Objetivos de Aprendizado da Aula

- Operar em Pilhas;
- Notação Polonesa Reversa;
- Gerenciar Ponteiros de Memória.

Conteúdo(s): Gerenciamento de Memória.



Máquina Virtual do Z0

Para o desenvolvimento do Z0 será adotada uma Máquina Virtual como fase intermediária. Essa máquina virtual pode ser vista como:

- Uma máquina com seu próprio ambiente (um computador virtual).
- Uma representação conveniente de um programa de computador entre o código de alto nível e a linguagem de máquina.

Comandos Suportados pela VM do Z0

Comandos Aritméticos e Booleanos

add (soma)
sub (subtração)
neg (negação)
eq (igual)
gt (maior que)
lt (menor que)
and (e)
or (ou)
not (não)

Comandos de Acesso de Memória

push segmento x (empilha)
pop segmento x (desempilha)

Comandos de Fluxo de Execução

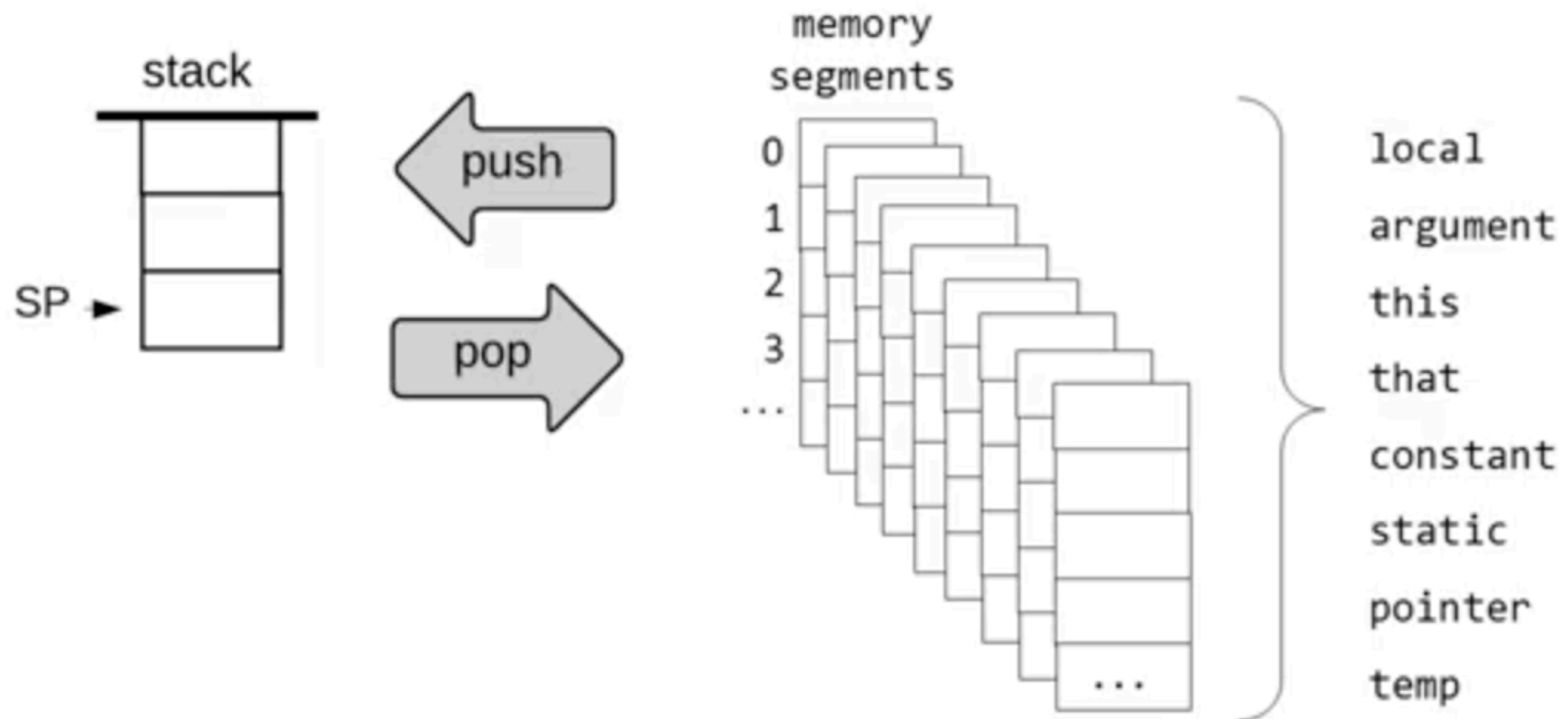
label (declaração)
goto (marcadores)
if-goto (marcadores)

Comandos de Chamada de Funções

function (declaração da função)
call (invoca uma função)
return (retorno da função)

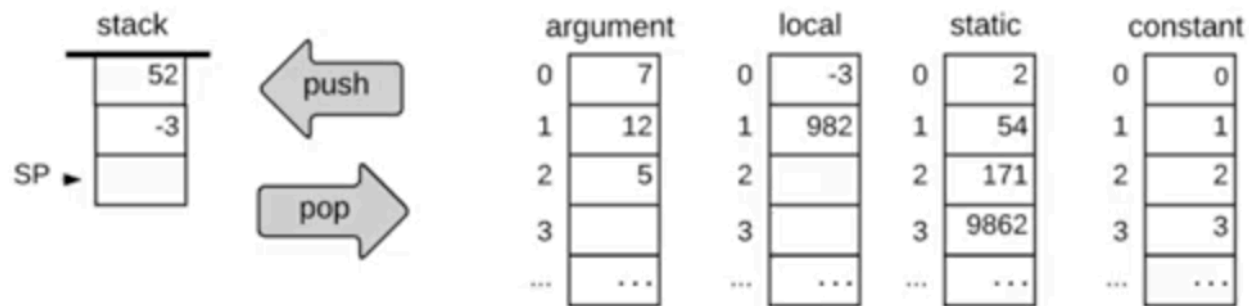
Segmentos de Memória

O usuário pode fazer o PUSH ou POP para algum segmento de memória.



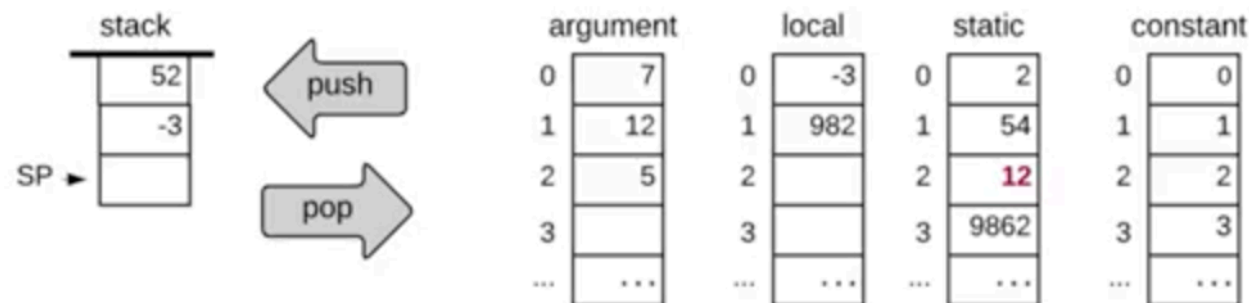
Pergunta

Qual é o código que realiza a operação abaixo?

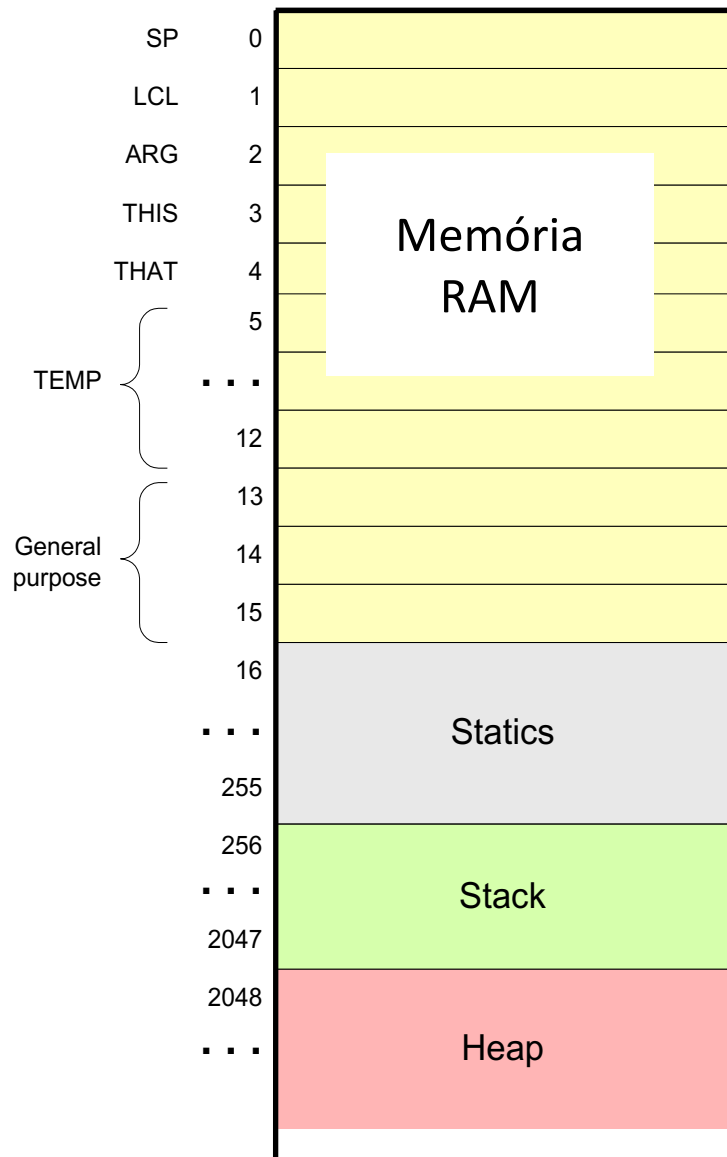


let static 2 = argument 1

?



Organização de memória na VM do Z0



- O **SP** (**S**tack **P**ointer) aponta a posição para a inserção de um novo elemento na pilha (Stack), ele é definido no endereço 0 (Zero) da RAM
- **LCL** (Local), **ARG** (Arguments), **THIS** e **THAT** são ponteiros usados para controlar a posição dos dados nas chamadas de funções.
- A região **Statics** é uma área de memória compartilhada por todas as funções.
- O **Stack** é a pilha de dados do computador. O espaço de memória entre os endereços 256 e 2047 é reservado para armazenar os dados da pilha.
- O **Heap** é a memória dinâmica de dados e pode ser acessada de diversas formas. Essa memória é reservada entre os endereços 2048 e 16383.

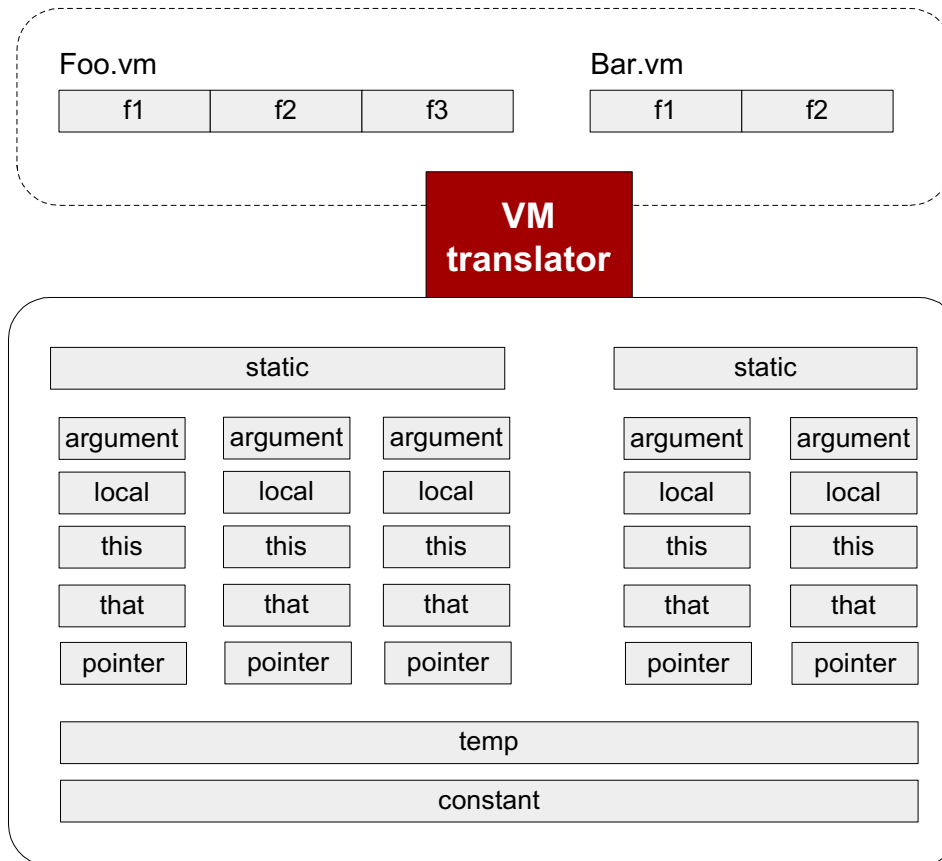
VM Emulator

The screenshot shows the Virtual Machine Emulator (1.4b3) interface. The window title is "Virtual Machine Emulator (1.4b3) - G:\examples\add". The menu bar includes File, View, Run, and Help. The toolbar contains icons for file operations, execution (step, run, break), and animation (slow, fast). The main area is divided into several panels:

- Program:** A list of instructions with addresses 0 to 14. The instruction at address 11 is highlighted in yellow.
- Static:** A panel for static variables.
- Local:** A panel for local variables, currently showing three slots with values 15, 8, and 0.
- Argument:** A panel for arguments.
- This:** A panel for the 'this' pointer.
- That:** A panel for the 'that' pointer.
- Temp:** A panel for temporary variables, currently showing two slots with values 0 and 0.
- Stack:** A panel showing the current stack with values 15 and 8. It is labeled "working stack".
- Call Stack:** A panel showing the call stack with entries: Sys.init, Main.main, and Main.add.
- emulator controls:** A panel with buttons for "Slow" and "Fast", and dropdowns for "Animate" (Program flow), "View" (Script), and "Format" (Decimal).
- virtual memory segments:** A panel showing the virtual memory segments, including "global stack" and "host RAM".
- default test script:** A panel showing the default test script, which is a "repeat" loop containing a "vmstep" instruction.
- Global Stack:** A panel showing the global stack with addresses 264 to 278 and values 0, 0, 15, 8, 0, 15, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0.
- RAM:** A panel showing the RAM with addresses 0 to 14 and values 0, 271, 266, 261, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0. A note indicates that the RAM is not part of the VM.

Organização dos arquivos

Vários arquivos .vm com suas funções gerando um só .asm



Manipulação de Ponteiros

Pseudocódigo:

$D = *p$

* significa a notação para a localização na memória apontada por p

Em Assembly:

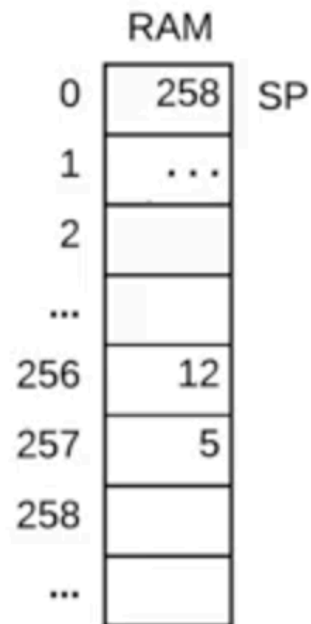
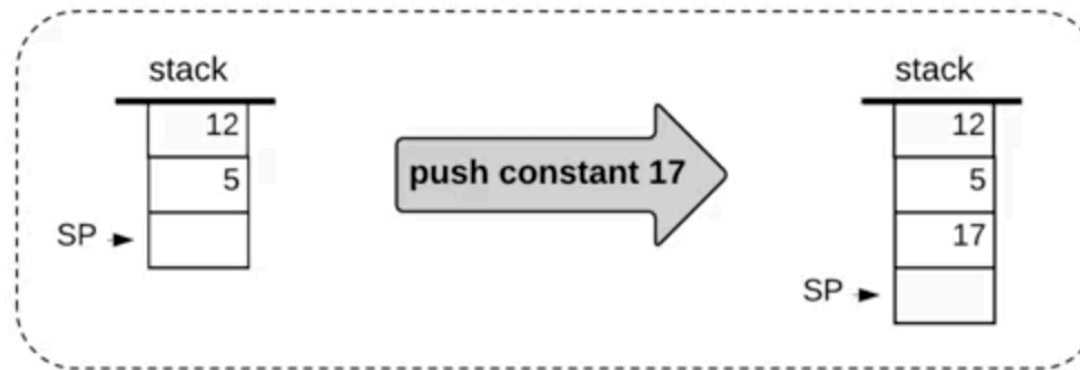
```
leaw $p,%A
movw (%A),%A
movw (%A),%D
```

Resultado:

D recebe o valor 23.

RAM		
0	257	p
1	1024	q
2	1765	
...	...	
256	19	
257	23	
258	903	
...	...	
1024	5	
1025	12	
1026	-3	
...	...	

Conversão das instruções



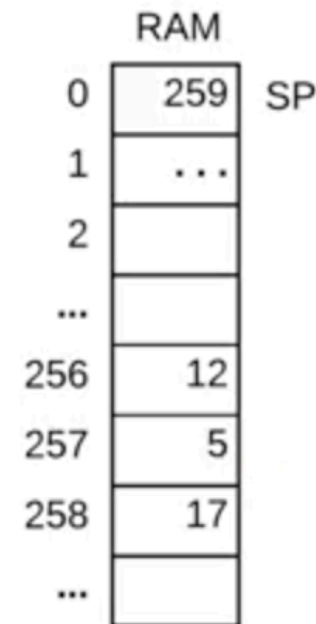
Lógica:

*SP = 17

SP++

Assembly:

```
leaw $17,%A
movw %A,%D
leaw $SP,%A
movw (%A),%A
movw %D, (%A)
leaw $SP,%A
movw (%A),%D
incw %D
movw %D, (%A)
```





Funções e Memória

Os programas na máquina virtual são organizados em funções, com cada função organizando seus dados e formas de acesso.

function NomeDaFunção nLocals

call NomeDaFunção nArgs

return

VEREMOS MAIS TARDE



Dúvidas ?

Lembrem-se:

Tragam sua dúvidas anotadas;

Verifiquem sites como:

- Google
- Stack Overflow
- Etc...

Use o Slack, para perguntar para seus colegas, ninjas e o professor antes da aula

Formar Duplas

Os Srs. Dennis Ritchie e Ken Thompson estão avaliando criar uma nova linguagem usando um máquina a pilha como linguagem intermediária. Você pode ajudar eles.

Formem duplas para resolver as atividades.

Mantenha um ambiente em que todos participem de tudo.



Exercício

1) Escreve um programa usando as instruções da máquina virtual do Z0 para avaliar qual dos 3 valores nas posições temporárias de memória é o maior, escrevendo ele na posição temp 3 :

```
push constant 3  
pop temp 0  
push constant 6  
pop temp 1  
push constant 9  
pop temp 2
```

Escreva o programa de forma que seja possível trocar os valores e o resultado continuar sendo correto.



Exercício

2) Implemente as seguintes instruções da máquina virtual em Assembly do Z0:

a) `push constant 54`

b) `pop temp 0`

c) `and`

d) `add`

Próxima Aula

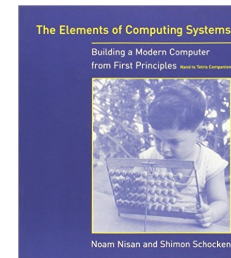
- Ver estudo para aula ? sobre ????
- Estudar Lista de Exercícios Aula ? (opcional):
- Ler (opcional)

Capítulo 8

The Elements of Computing Systems

Building a Modern Computer from First Principles

Noam Nisan e Shimon Schocken



Insper

www.insper.edu.br