CPSC-240 Computer Organization and Assembly Language

Chapter 9

Process Stack

Instructor: Yitsen Ku, Ph.D.
Department of Computer Science,
California State University, Fullerton, USA

CALIFORNIA STATE UNIVERSITY
FULLERTON™

Outline

- Stack Example
- Stack Instructions
- Stack Implementation
- Stack Layout
- Stack Operations
- Stack Example

Stack Example

Stack Example

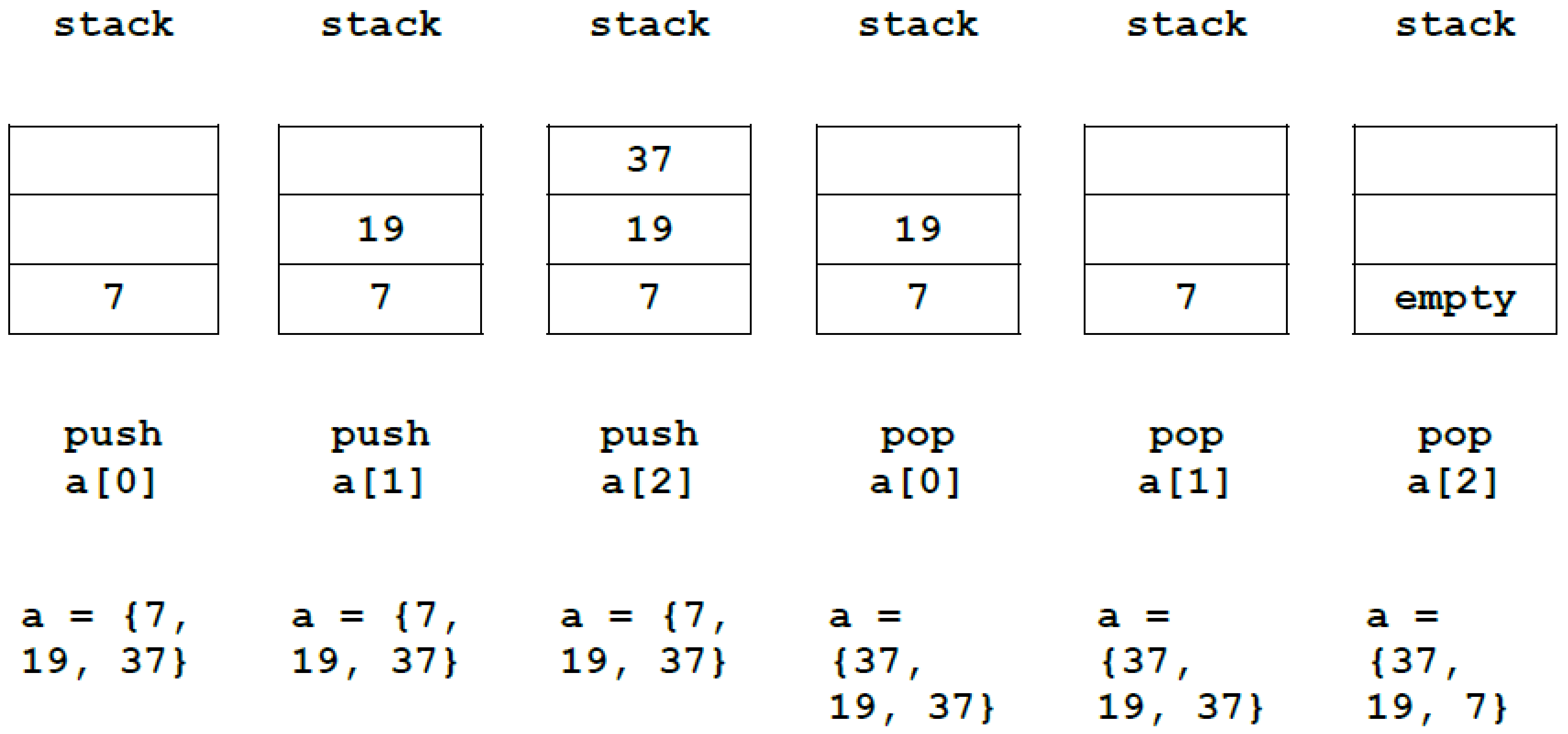
- a = {7, 16, 37}
- push operations:

```
        push    a[0]            // rsp-=8, [rsp] = a[0] = 7
        push    a[1]            // rsp-=8, [rsp] = a[1] = 19
        push    a[2]            // rsp-=8, [rsp] = a[2] = 37
```

- pop operations:

```
        pop     a[0]            // a[0] = [rsp] = 37, rsp+=8
        pop     a[1]            // a[1] = [rsp] = 19, rsp+=8
        pop     a[2]            // a[2] = [rsp] = 7, rsp+=8
```

Stack Example



Stack Instructions

Stack Instructions

- A push operation puts things onto the stack, and a pop operation takes things off the stack. The format for these commands is:

```
push  <operand64>
pop   <operand64>
```

- The operand can be a register or memory, but an immediate is not allowed.

Stack Instructions

Instruction	**Explanation**
`push    <op64>`	Push the 64-bit operand on the stack. First, adjusts **rsp** accordingly (**rsp**-8) and then copy the operand to [**rsp**]. The operand may not be an immediate value. Operand is not changed.

Examples:	`push   rax`
	`push   qword [qVal]  ; value`
	`push   qVal          ; address`

`pop    <op64>`	Pop the 64-bit operand from the stack. Adjusts **rsp** accordingly (**rsp**+8). The operand may not be an immediate value. Operand is overwritten.

Examples:	`pop   rax`
	`pop   qword [qVal]`
	`pop   rsi`

Stack Implementation

Stack Implementation

- The **rsp** register is used to point to the current top of stack in memory.
- In this architecture, as with most, the stack is implemented growing downward in memory.

Stack Layout

CALIFORNIA STATE UNIVERSITY FULLERTON

The General Memory Layout for a Program

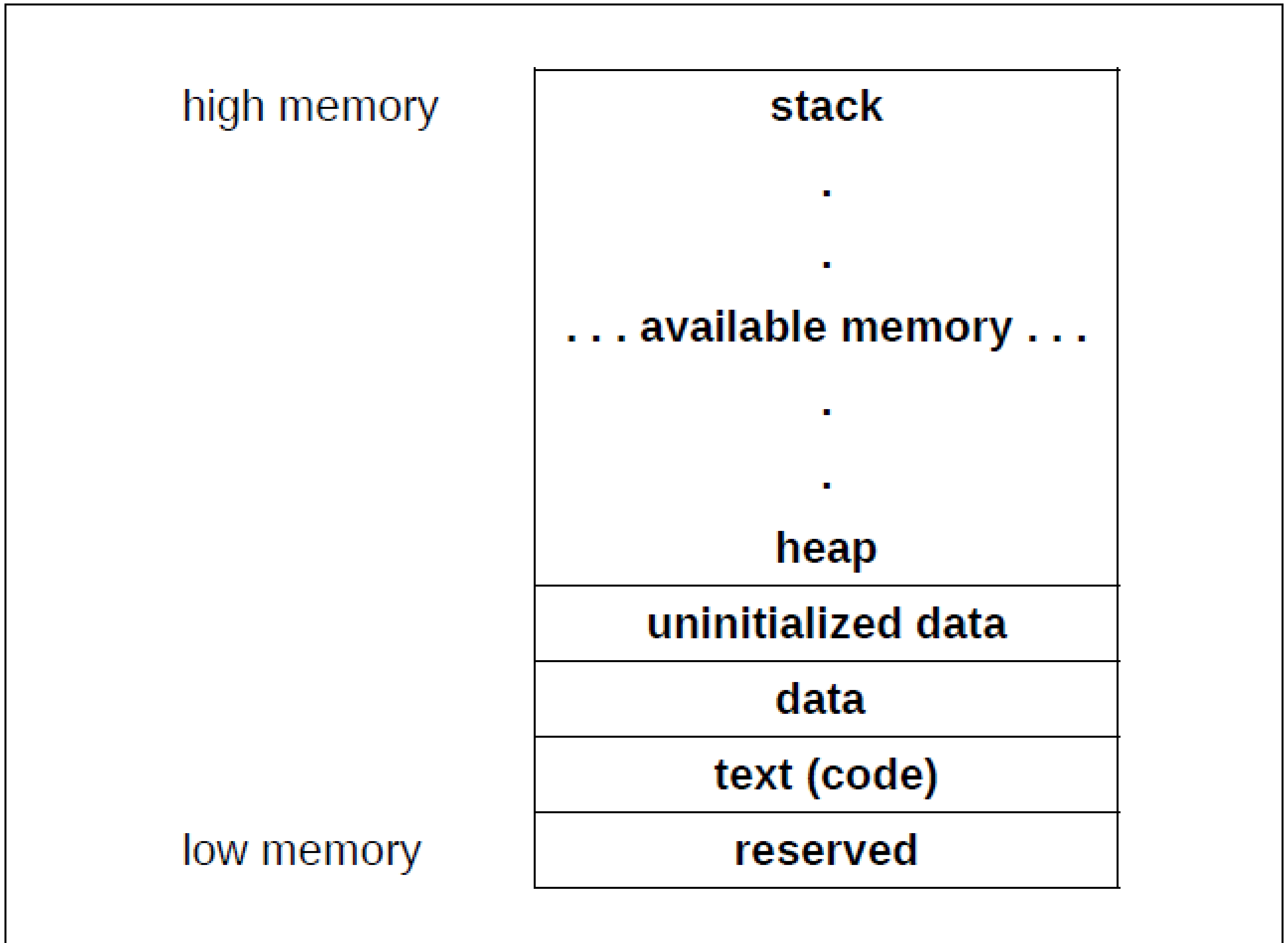


Illustration 22: Process Memory Layout

Stack Layout

- As the heap and stack expand, they grow toward each other. This is done to ensure the most effective overall use of memory.
- A program (Process A) that uses a significant amount of stack space and a minimal amount of heap space will function.
- A program (Process B) that uses a minimal amount of stack space and a very large amount of heap space will also function.

CALIFORNIA STATE UNIVERSITY FULLERTON

Stack Layout

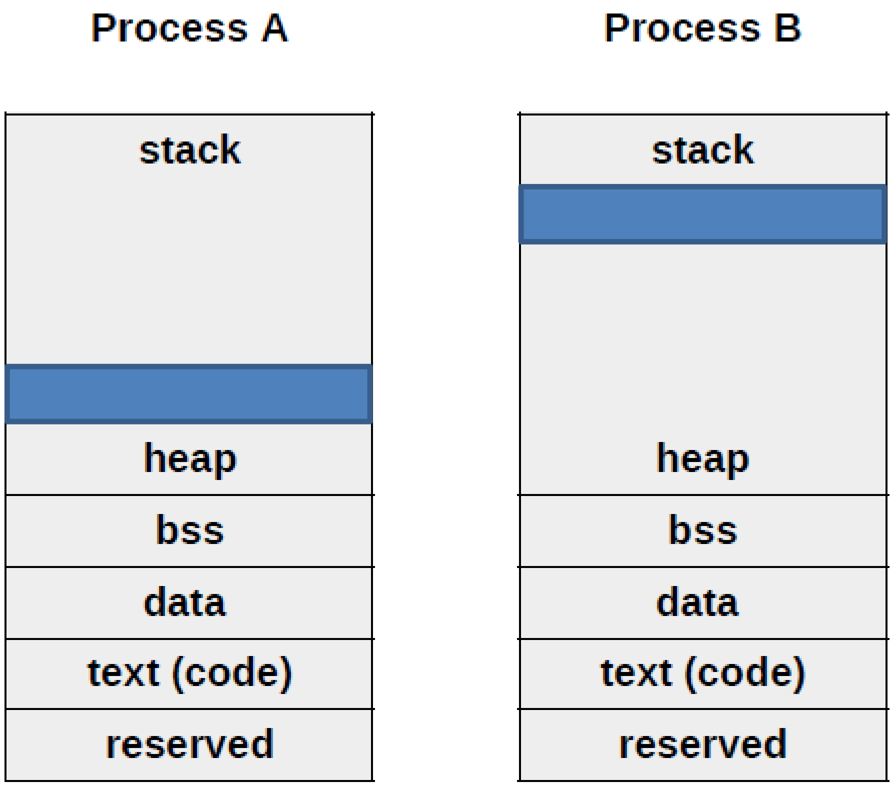


Illustration 23: Process Memory Layout Example

Stack Operations

Stack Operations

For a push operation:

1. The **rsp** register is decreased by 8 (1 quadword).
2. The operand is copied to the stack at **[rsp]**.

- The operand is not altered. The order of these operations is important.

For a pop operation:

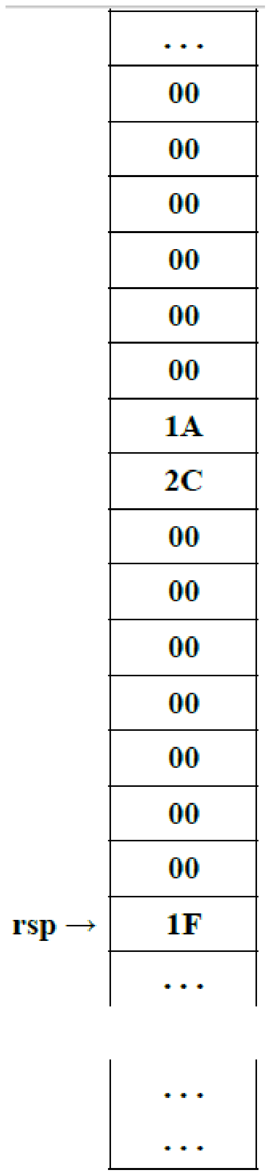
1. The current top of the stack, at **[rsp]**, is copied into the operand.
2. The **rsp** register is increased by 8 (1 quadword).

- The order of these operations is the exact reverse of the push.

Stack Operations

Ex:

```
mov     rax, 6700                ; 6700₁₀ = 00001A2C₁₆
push    rax
mov     rax, 31                  ; 31₁₀ = 0000001F₁₆
push    rax
```


. . .
00
00
00
00
00
00
1A
2C
00
00
00
00
00
00
00
rsp →
1F
. . .
. . .
. . .

Stack Example

Stack Example (1)

```
; Simple example demonstrating basic stack operations.
; Reverse a list of numbers - in place.
; Method: Put each number on stack, then pop each number
; back off, and then put back into memory.
;
*****************************************************
; Data declarations
section         .data
; -----
; Define constants
EXIT_SUCCESS equu    0          ; successful operation
SYS_exit      equ    60         ; call code for terminate
```

Stack Example (2)

```
; -----
; Define Data.
numbers         dq          121, 122, 123, 124, 125
len             dq          5
;
*******************************************************

section         .text
global _start
_start:
; Loop to put numbers on stack.
        mov     rcx, qword [len]
        mov     rbx, numbers
        mov     r12, 0
        mov     rax, 0
```

CALIFORNIA STATE UNIVERSITY FULLERTON

Stack Example (3)

```
pushLoop:
        push    qword [rbx+r12*8]
        inc     r12
        loop    pushLoop
; -----
; All the numbers are on stack (in reverse order).
; Loop to get them back off. Put them back into
; the original list...
        mov     rcx, qword [len]
        mov     rbx, numbers
        mov     r12, 0
```

Stack Example (4)

```
popLoop:
        pop     rax
        mov     qword [rbx+r12*8], rax
        inc     r12
        loop    popLoop
; -----
; Done, terminate program.
last:
        mov     rax, SYS_exit           ; call code for exit
        mov     rdi, EXIT_SUCCESS       ; exit with success
        syscall
```

CALIFORNIA STATE UNIVERSITY FULLERTON

Thanks