Chapter 04: Instruction Sets and the Processor organizations

Lesson 15:

Stacks Addressing

Objective

- **To learn stacks and stack operations**

Stack

Stack

1. Stack consists of a set of locations, each of which can hold one word of data
2. When a value adds to a stack, it is placed in the *top* location of the stack, and all data currently in the stack moves down one location with respect to stack top

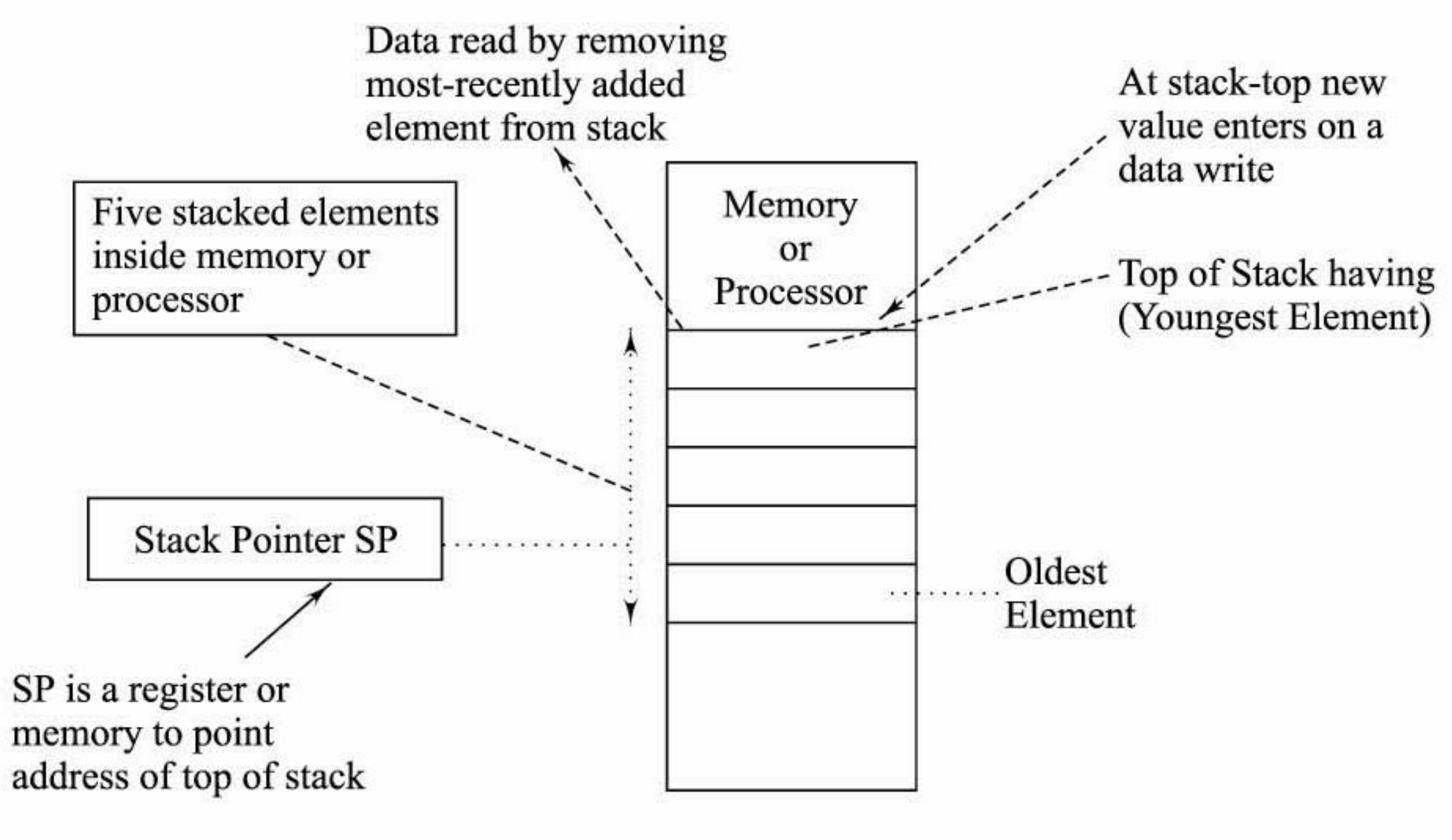
Stack

3. Data can only be removed from the top of the stack

- When this is done, all other data in the stack moves up one location

4. In general, data cannot be read from a stack without disturbing the stack, although some processors may provide special operations to allow this

Stack



Data read by removing most-recently added element from stack
At stack-top new value enters on a data write
Five stacked elements inside memory or processor
Memory or Processor
Top of Stack having (Youngest Element)
Stack Pointer SP
Oldest Element
SP is a register or memory to point address of top of stack

A stack data structure

- A last-in-first-out (LIFO) data structure
- The name *stack* comes from the fact that the data structure acts like a stack of plates. when a new plate is put on a stack of plates, it goes on the top, and it is the first plate removed when someone takes a plate off of the stack

Stack Basic Operations on a Stack

- **PUSH** operation— Takes one argument (element) and places the value of the argument (element) on the top of the stack, pushing all previous data down one location.
- **POP** operation — Removes the top value from the stack and returns it, allowing the value to be used as the input to an instruction.

Using Computer Memory for Implementing Stack Operations

- A fixed location defines the bottom of the stack, and a pointer (S0) gives the location of the top of the stack (the location of the last value pushed onto the stack)

Using Computer Memory for Implementing Stack Operations

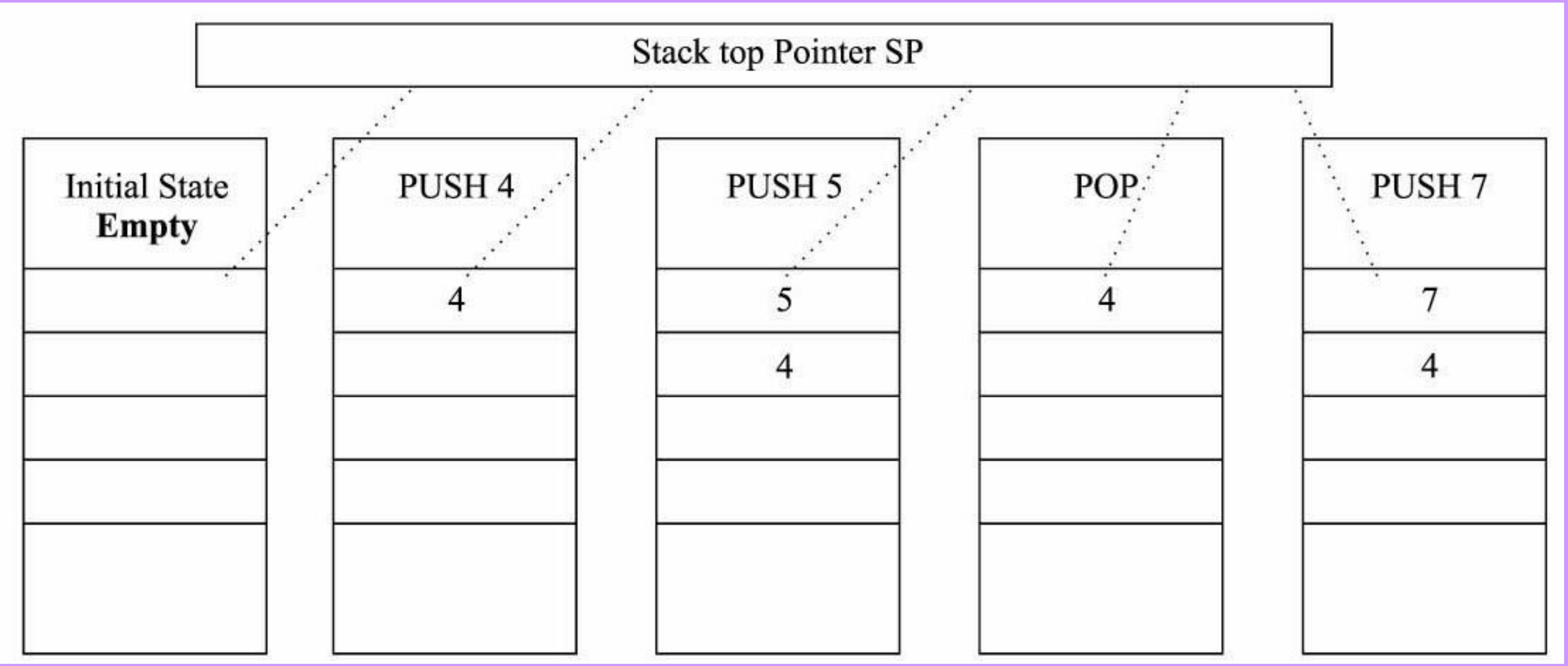
- A fixed location pointer (S0) defines the bottom of the stack, and a pointer (SP) gives the location of the top of the stack (the location of the last value pushed onto the stack)
- S0 = highest memory address available

An Approach when S0 at Highest address

- Several approaches possible
- Including ones where the bottom pointer (S0) points to the highest address in the stack buffer and the stack grows toward lower addresses
- Results in a completely functional stack, but accessing the stack tends to be relatively slow, because of the latency of the memory system

Push and Pop operation on Stack

A set of Push and Pop instructions affecting the stack



Size of a Stack

- As an abstract data structure, stacks are assumed to be infinitely deep, meaning that an arbitrary amount of data can be placed on the stack by the program
- In practice, stacks are implemented using buffers in memory, which are finite in size
- If the amount of data in the stack exceeds the amount of space allocated to the stack, *overflow* error occurs

Example using r13 as stack pointer

- Assume— 32-bit words in memory
- Move immediate #0x00FFFF, an address in the memory for stack top
- Now place r0 to r2 on to stack

Address of stack top after the operations when using r13 for stack top

- MOV r13, #0x00FFFF
- PUSH r0; /* r13 will be r13 – 4 = 0x00FFFB*/
- PUSH r1; /* r13 will be r13 – 4 = 0x00FFF7*/
- PUSH r2; /* r13 will be r13 – 4 = 0x00FFF3*/

Summary

We Learnt

- Stacks
- Stack pointer SP
- Top of stack
- Push and pop operations on stack

End of Lesson 15on

Stacks Addressing