Chapter 04: Instruction Sets and the Processor organizations

Lesson 16:

Queues Addressing

Objective

- **To learn queues and their addressing**

Queue

Queue

1. Queue consists of a set of locations, each of which can hold one word of data
2. When a value adds to a Queue, it is placed in the *tail* location of the Queue, QTail and all data currently in the Queue moves up one location

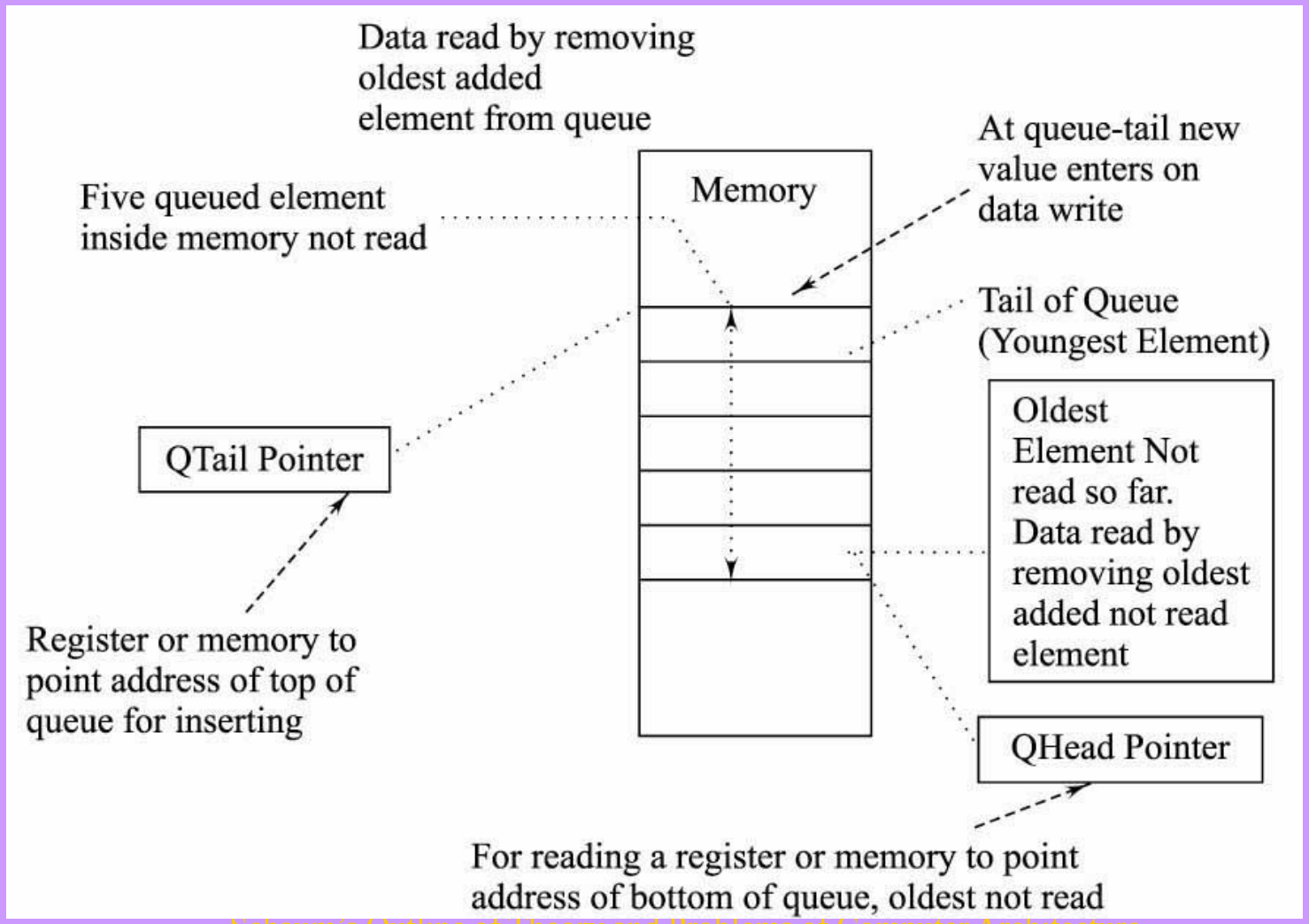
Queue

3. Data can only be removed (deleted) from the head of the Queue
- When this is done, all other data in the Queue moves down one location
4. Data can only be inserted at the tail of the Queue
- When this is done, all other data in the Queue moves up one location

A Queue data structure

- A first-in-first-out (FIFO) data structure
- The name *Queue* comes from the fact that the data structure acts like a Queue of characters for printing in a memory
- When a new character *c* adds in a Queue of characters, the *c* goes on the tail, and that is the last character printed when printer prints the characters in the queue

Queue



Basic Operations Insert and Delete on a Queue

Basic Operations on a Queue

- **Insert** operation— Takes one argument (element) and places the value of the argument (element) on the tail of the Queue, Inserting results in all previous data up one location
- **Delete** operation— Removes the head value from the Queue and returns it, allowing the value to be used as the input to an instruction

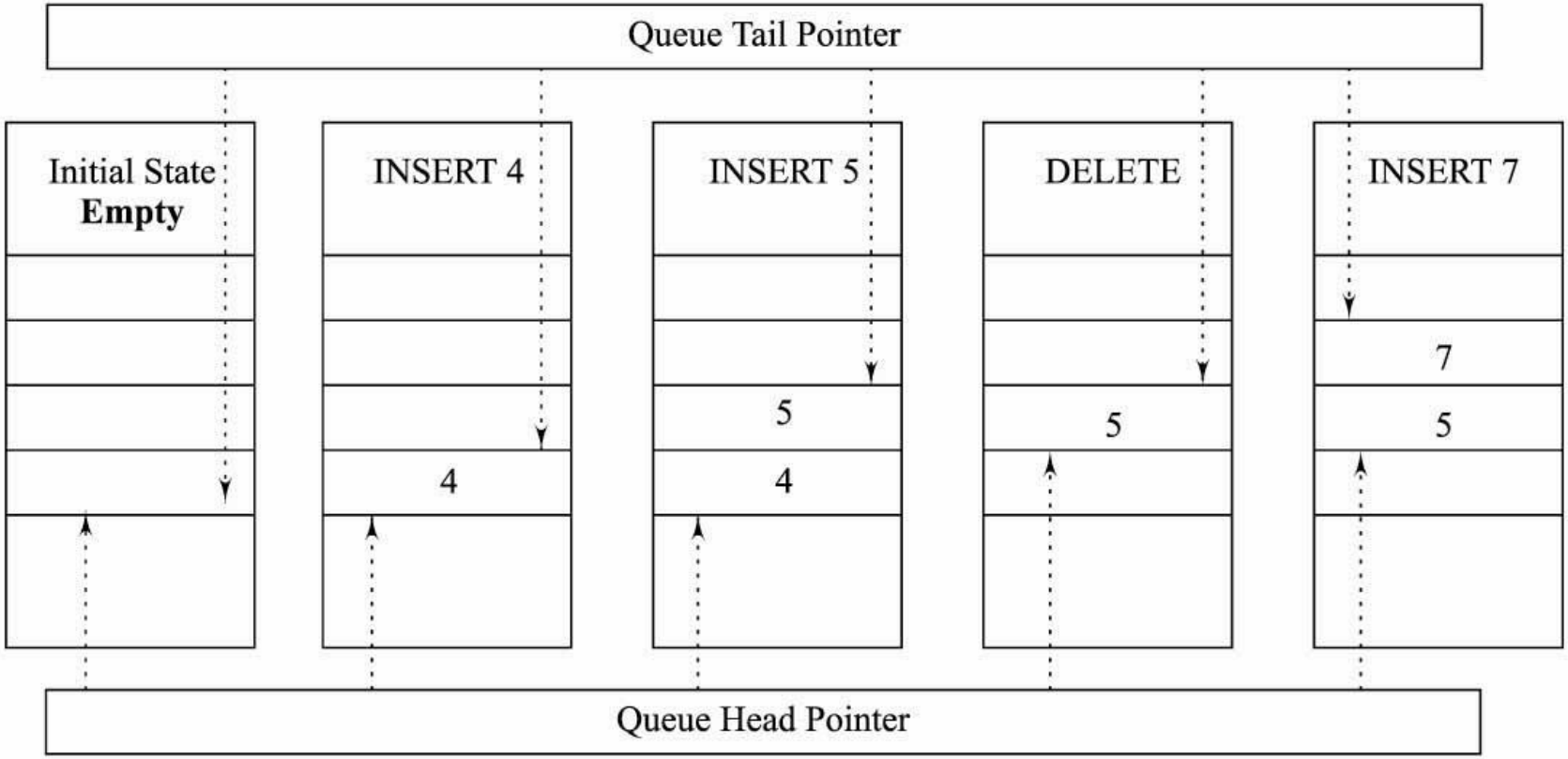
Using Computer Memory for Implementing Queue Operations

- A fixed location Q0 defines the bottom of the Queue
- A tail pointer (QTail) gives the location of the tail of the Queue (the location of the last value inserted onto the Queue)
- A head pointer (QHead) gives the location of the head of the Queue (the location of the first value when deleted from the Queue

An Approach when Q0 at Lowest address

- Several approaches are possible
- Including ones where the bottom pointer (Q0) points to the lowest address in the Queue buffer and the Queue grows toward higher addresses
- This approach results in a completely functional Queue
- But accessing the Queue tends to be relatively slow, because of the latency of the memory system

Queue in Memory



Size of a Queue

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

Example of Operations on Queue

Example r0 to r2 numbers in ascending order

- Use r7 and r8 as queue tail and queue head pointers
- The r0 to r2 have the numbers in ascending order
- Now Arranging them in same order in memory using a queue

Example r0 to r2 numbers in ascending order

- Use 0x001000 as address in the memory for queue tail and queue head at the star
- Assume─ 32-bit words in memory

MOV r7, #0x001000 / * Tail Pointer */

MOV r8, #0x001000 / * Head Pointer */

ST (r7)+, r0 /* r0 → memory 0x001000, tail pointer r7 will become r7 + 4 = 0x001004*/

ST (r7)+, r1 /* r1 → memory 0x001004, tail pointer r7 will become r7 + 4 = 0x001008*/

Example r0 to r2 numbers in ascending order

ST (r7)+, r1 /* r1 → memory 0x001008, tail pointer r7 will become r7 + 4 = 0x00100C*/

The r7 will have the address 0x00100C for queue tail after the operation, but queue head unchanged because there is no deletion from queue

Summary

We Learnt

- Queues
- queue-head
- queue-tail pointers
- Using QHead and QTail in Insert and delete operations on the queue

End of Lesson 16 on
Queues Addressing