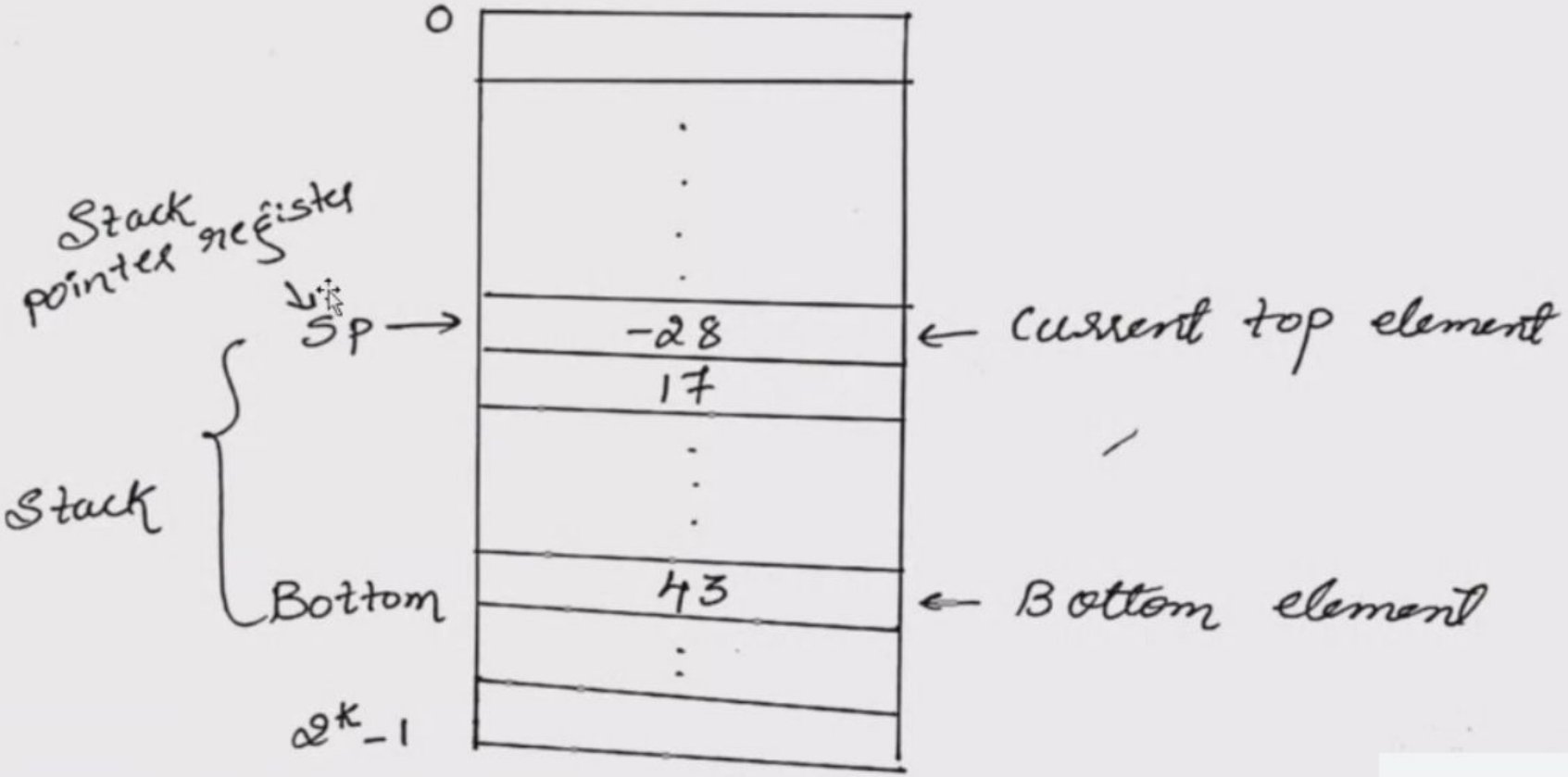

0
Stack pointer register
SP
-28
17
43
Bottom
$2^k - 1$
Stack
current top element
Bottom element

Queues

- https://www.youtube.com/watch?v=UpvDOm3prfI

You

Subroutines

- **SUBROUTINES**
- A subtask consisting of a set of instructions which is executed many times is called a **Subroutine**.
- A Call instruction causes a branch to the subroutine (Figure: 2.16).
- At the end of the subroutine, a return instruction is executed
- Program resumes execution at the instruction immediately following the subroutine call
- The way in which a computer makes it possible to call and return from subroutines is referred to as its **Subroutine Linkage** method.
- The simplest subroutine linkage method is to save the return-address in a specific location, which may be a register dedicated to this function. Such a register is called the **Link Register**.
- When the subroutine completes its task, the Return instruction returns to the calling-program by branching indirectly through the link-register.
- The **Call Instruction** is a special branch instruction that performs the following operations:
- → Store the contents of PC into link-register.
- → Branch to the target-address specified by the instruction.
- The **Return Instruction** is a special branch instruction that performs the operation:
- → Branch to the address contained in the link-register.
- https://www.youtube.com/watch?v=ADl6mYc7Uk4
-

Click to add title

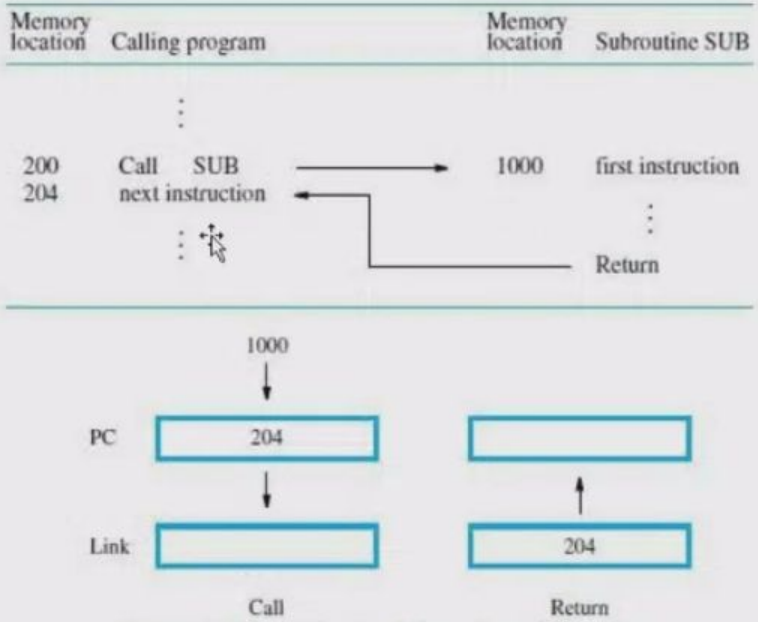


Figure 2.16 Subroutine linkage using a link register.

You

Click to add title

- **SUBROUTINE NESTING AND THE PROCESSOR STACK**
- **Subroutine Nesting** means one subroutine calls another subroutine.
- In this case, the return-address of the second call is also stored in the link-register, destroying its previous contents.
- Hence, it is essential to save the contents of the link-register in some other location before calling another subroutine. Otherwise, the return-address of the first subroutine will be lost.
- Subroutine nesting can be carried out to any depth. Eventually, the last subroutine called completes its computations and returns to the subroutine that called it.
- The return-address needed for this first return is the last one generated in the nested call sequence. That is, return-addresses are generated and used in a LIFO order.
- This suggests that the return-addresses associated with subroutine calls should be pushed onto a stack. A particular register is designated as the SP(Stack Pointer) to be used in this operation.
- SP is used to point to the processor-stack.
- Call instruction pushes the contents of the PC onto the processor-stack.
- Return instruction pops the return-address from the processor-stack into the PC.

You

Additional instructions

- **LOGIC INSTRUCTIONS**
- Logic operations such as AND, OR, and NOT applied to individual bits.
- These are the basic building blocks of digital-circuits.
- This is also useful to be able to perform logic operations is software, which is done using instructions that apply these operations to all bits of a word or byte independently and in parallel.

You

Click to add title

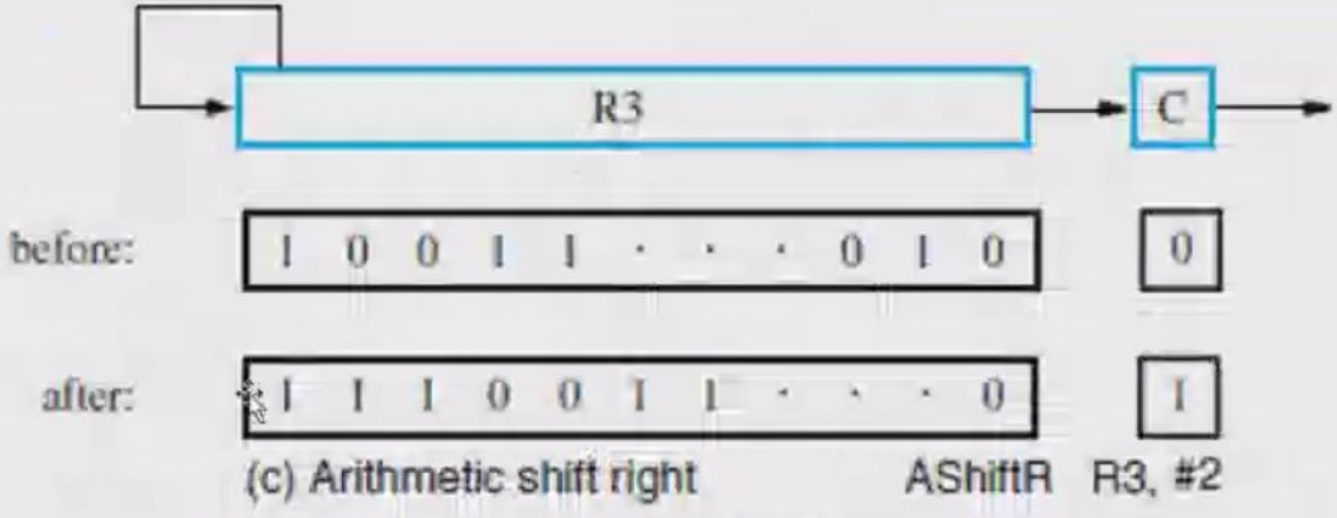
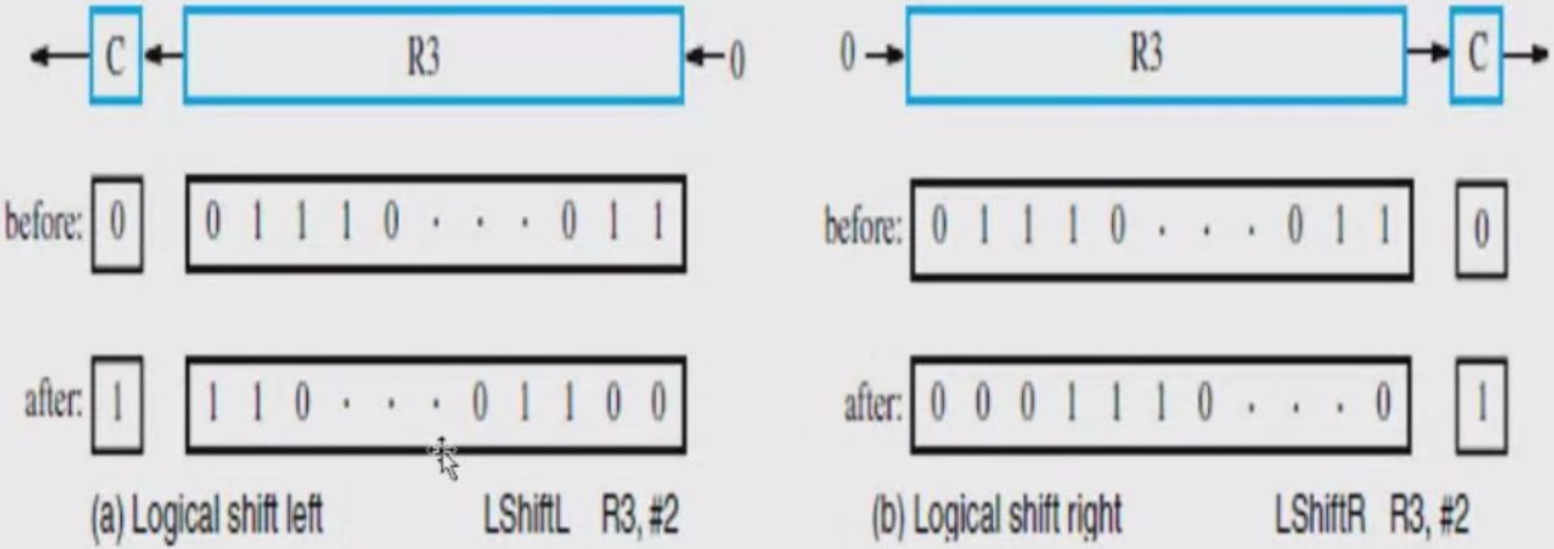


Figure 2.23 Logical and arithmetic shift instructions.

You

Click to add title



Click to add title

- LshiftL count,destination
- 0 01110.....011
- LshiftL #2,R0
- 0 1110.....0110
- 1 110.....01100

You

Click to add title

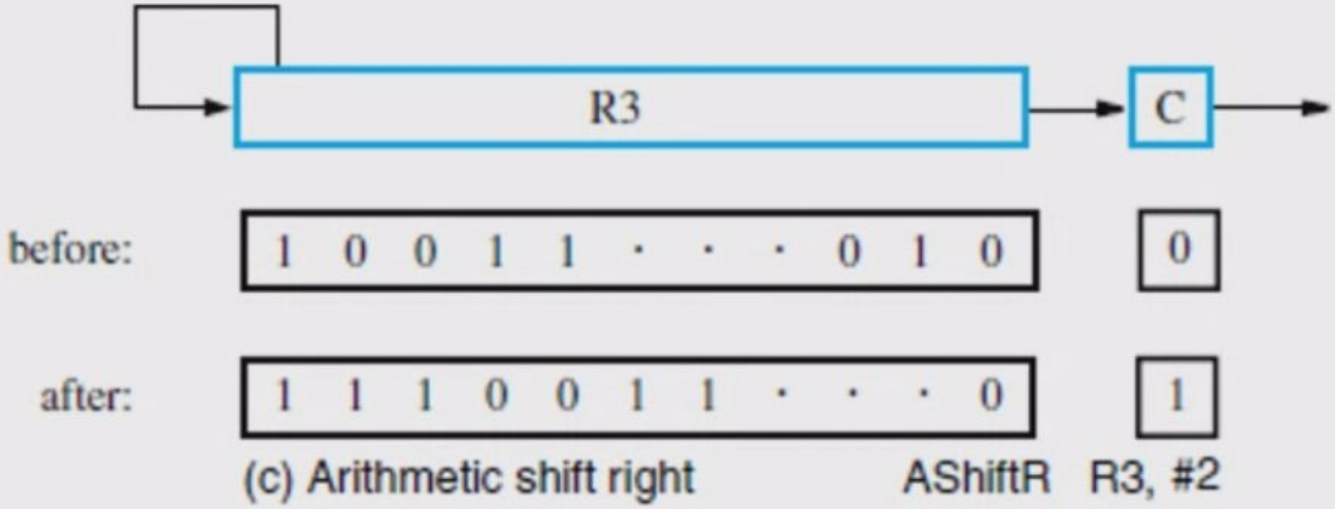


Figure 2.23 Logical and arithmetic shift instructions.

You

Click to add title

- **ROTATE OPERATIONS**
- In shift operations, the bits shifted out of the operand are lost, except for the last bit shifted out which is retained in the Carry-flag C.
- To preserve all bits, a set of rotate instructions can be used.
- They move the bits that are shifted out of one end of the operand back into the other end.
- Two versions of both the left and right rotate instructions are usually provided. In one version, the bits of the operand is simply rotated.
- In the other version, the rotation includes the C flag.
-

You

Click to add title

- Digit packing
- Unpacked BCD
- 2

8	8	8	0000 0010

- Packed BCD

You

Click to add title

- Digit packing
- Unpacked BCD
- 2

8	8	8	0000 0010

- Packed BCD 4bits

Loc	4	34	00110100
Loc+1	2	32	00110010

You

Click to add title

QUEUE

- Data are stored in and retrieved from a queue on a FIFO basis.
- Difference between stack and queue?

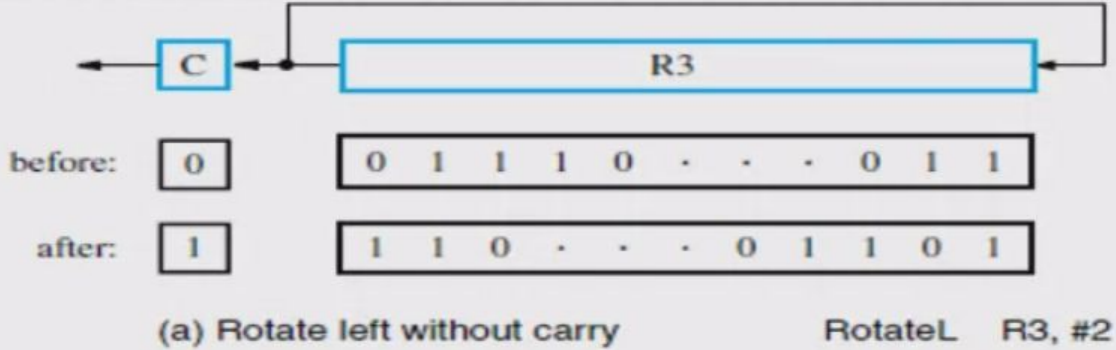
1) One end of the stack is fixed while the other end rises and falls as data are pushed and popped.

2) In stack, a single pointer is needed to keep track of top of the stack at any given time.

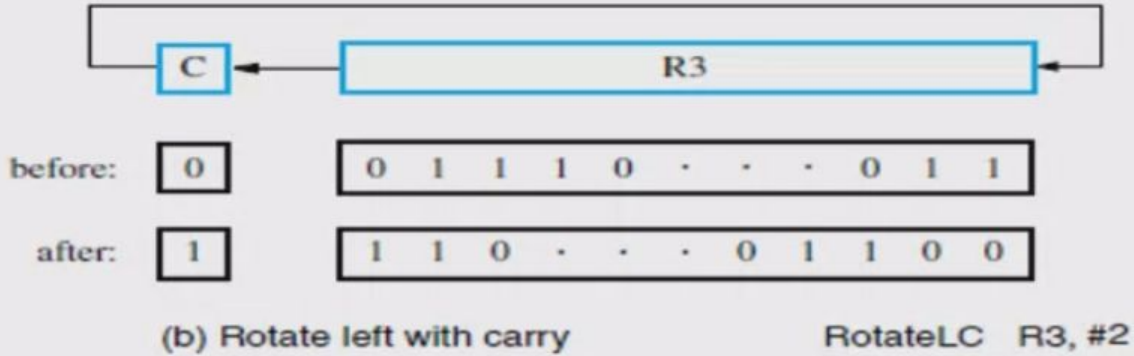
- In queue, two pointers are needed to keep track of both the front and end for removal and insertion respectively.

3) Without further control, a queue would continuously move through the memory of a computer in the direction of higher addresses. One way to limit the queue to a fixed region in memory is to use a circular buffer.

Click to add title



(a) Rotate left without carry RotateL R3, #2



(b) Rotate left with carry RotateLC R3, #2

ENCODING OF MACHINE INSTRUCTIONS

- To be executed in a processor, an instruction must be encoded in a binary-pattern. Such encoded instructions are referred to as **Machine Instructions**.
- The instructions that use symbolic-names and acronyms are called *assembly language instructions*.
- We have seen instructions that perform operations such as add, subtract, move, shift, rotate, and branch. These instructions may use operands of different sizes, such as 32-bit and 8-bit numbers.
- Let us examine some typical cases. The instruction
- *Add R1, R2* ;Has to specify the registers R1 and R2, in addition to the OP code. If the processor has 16 registers, then four bits are needed to identify each register. Additional bits are needed to indicate that the Register addressing-mode is used for each operand.
- The instruction
- *Move 24(R0), R5* ;Requires 16 bits to denote the OP code and the two registers, and some bits to express that the source operand uses the Index addressing mode and that the index value is 24.

You