Stacks

- A **stack data structure** can be used to store the return addresses associated with subroutine calls .
- Call-subroutine pushes the content of the PC onto the stack and loads the subroutine address into the PC .
- The return instruction pops the return address from the stack into the PC .

What is the stack?

- A memory block used to temporarily save values, beyond the amount of data that registers can hold
- *Push* adds a given node to the top of the stack leaving previous nodes below.
- *Pop* removes and returns the current top node of the stack.
- Typically grows towards descending addresses

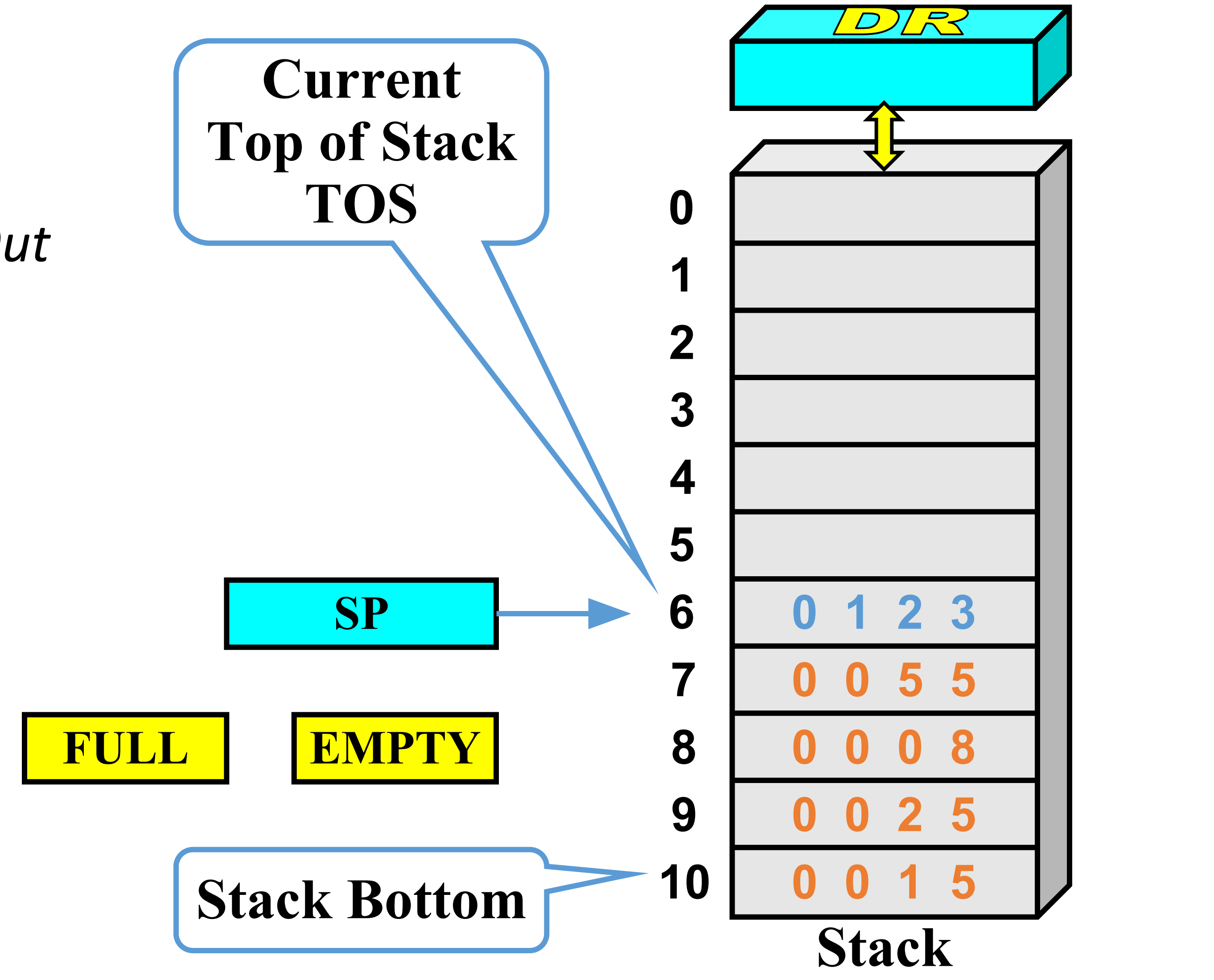
Home Work

- For each Addressing modes mentioned before, state one example for each addressing mode stating the specific benefit for using such addressing mode for such an application.

Stack Organization

- LIFO

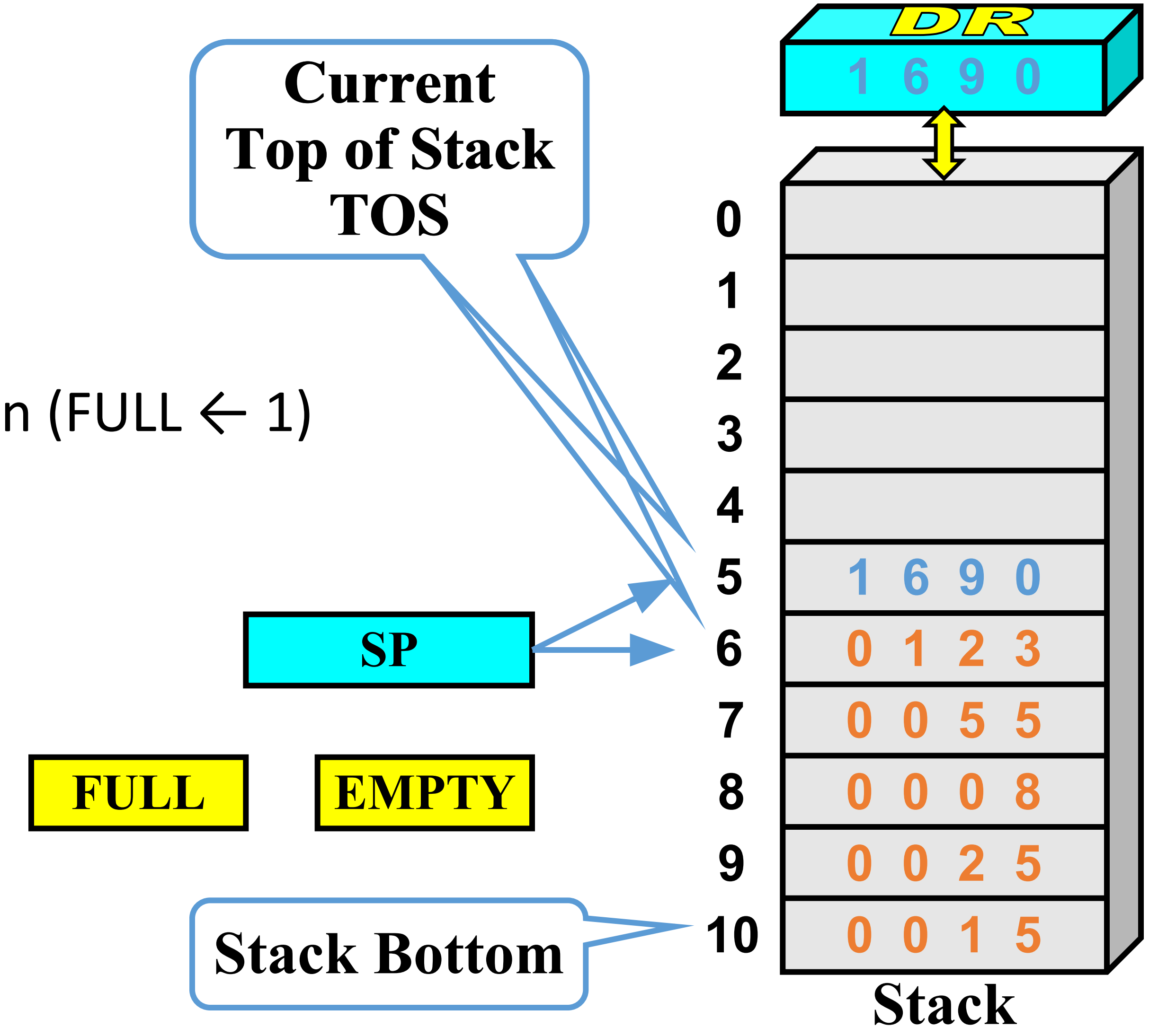
  Last In First Out



Stack Organization

- PUSH

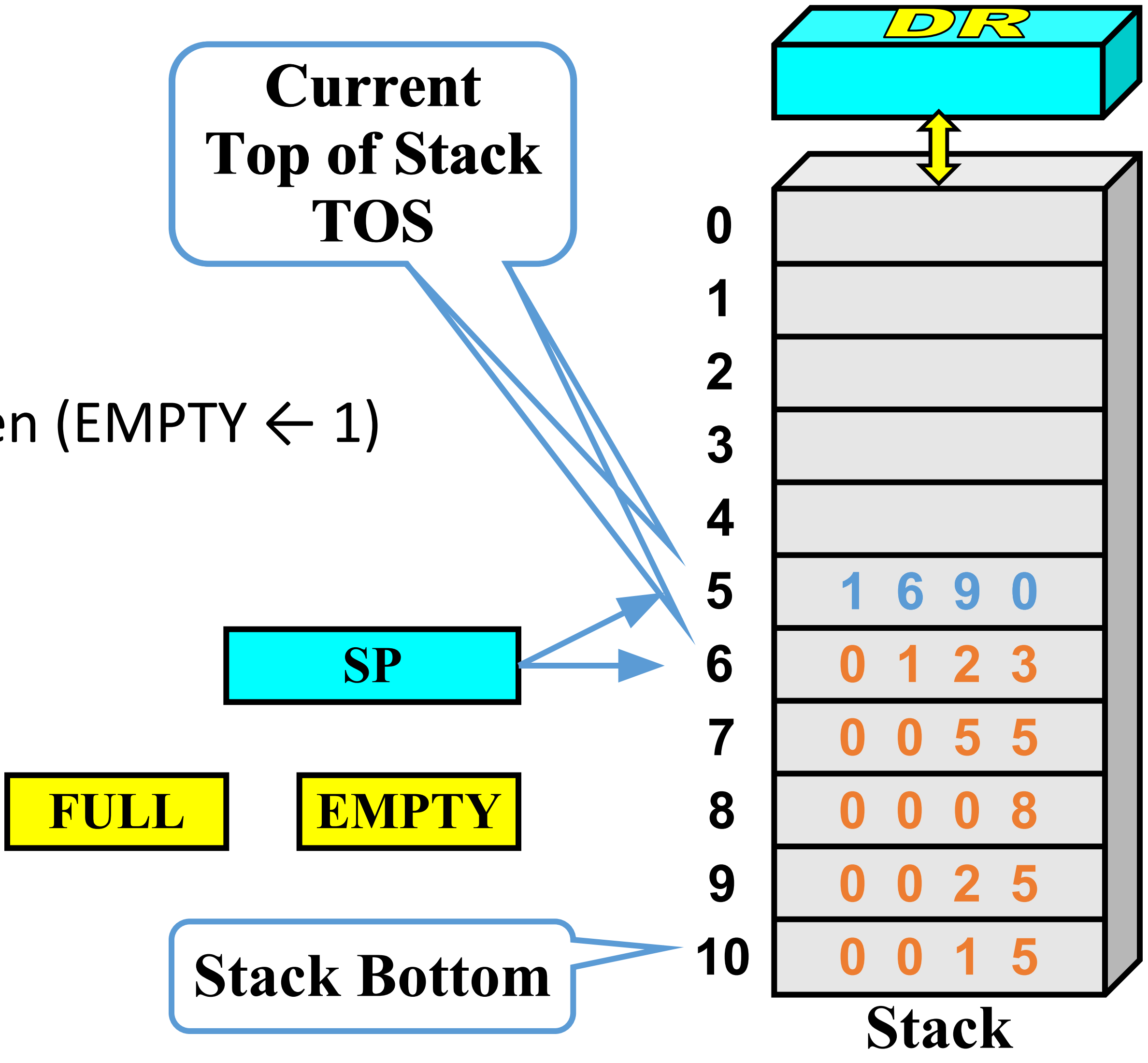
```
    SP ← SP – 1
    M[SP] ← DR
    If (SP = 0) then (FULL ← 1)
    EMPTY ← 0
```



Stack Organization

- POP

```
DR ← M[SP]
SP ← SP + 1
If (SP = 11) then (EMPTY ← 1)
FULL ← 0
```



Stack Organization

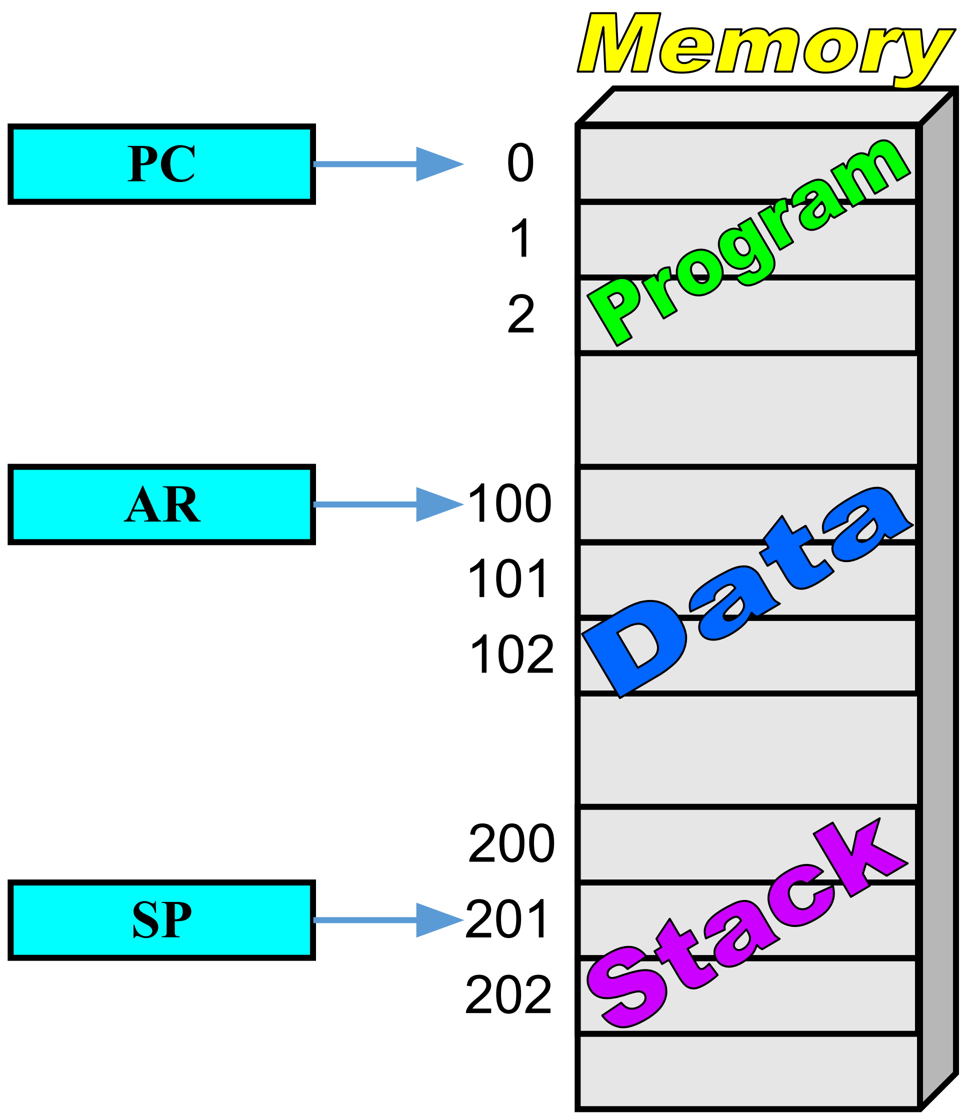
- Memory Stack
  - PUSH

    SP ← SP – 1

    M[SP] ← DR
  - POP

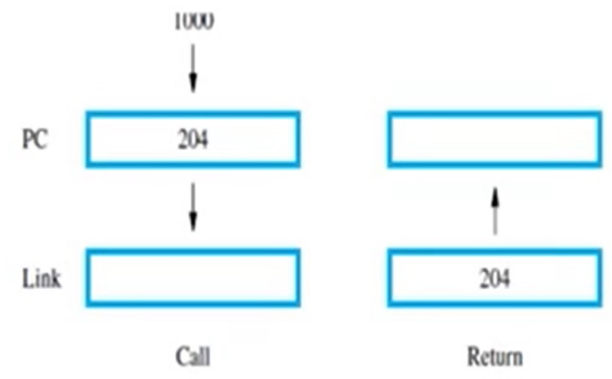
    DR ← M[SP]

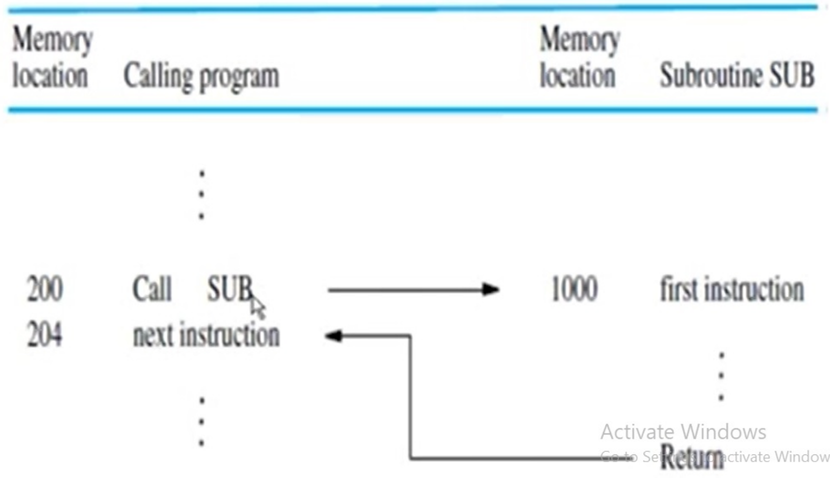
    SP ← SP + 1



- Call_subroutine is a special branch instruction that performs the following operations
  - store the contents of the PC in the link register (LR)
  - branch to the target address specified by the instruction
- The return from a subroutine branches to the address contained in the link register

Subroutine Linkage





Subroutine Call and Return

- ❑ It is a self-contained sequence of instructions that performs a given computational task.
- ❑ During the execution of a program,a subroutine may call when it is called, a branch is executed to the beginning of the subroutine to start executing its set of instructions. After the subroutine has been executed,a branch is made back to the main program.

 A subroutine call is implemented with the following microoperations:

CALL:

SP← SP-1**: Decrement stack point**

M[SP] ←PC **: Push content of PC onto the stack**

PC←*Effective Address* **: Transfer control to the subroutine**

RETURN:

PC ← M[SP] **: Pop stack and transfer to PC**

SP ← SP+1 **: Increment stack pointer**