# Stack and Subroutine in 8085

Microprocessor and Interfacing Techniques
(CS202)

B.Tech. II, Semester IV


Department of Computer Science and Technology
S.V.National Institute of Technology-Surat

January 30, 2023

# The Stack

- Stack is an area of memory for keeping temporary data.
- Stack is used by CALL instruction to keep return address for procedure,
- RET instruction gets this value from the stack and returns to that offset.
- Quite the same thing happens when INT instruction calls an interrupt, it stores in stack flag register, code segment and offset.
- IRET instruction is used to return from interrupt call.
- We can also use the stack to keep any other data,
- There are two instructions that work with the stack
  - PUSH - stores 16 bit value in the stack.
  - POP - gets 16 bit value from the stack.

# Syntax of PUSH instruction

PUSH REG
PUSH SREG
PUSH memory
PUSH immediate

- REG: AX, BX, CX, DX, DI, SI, BP, SP.
- SREG: DS, ES, SS, CS.
- memory: [BX], [BX+SI+7], 16 bit variable, etc...
- immediate: 5, -24, 3Fh, 10001101b, etc...

# Syntax of POP instruction

POP REG
POP SREG
POP memory

- REG: AX, BX, CX, DX, DI, SI, BP, SP.
- SREG: DS, ES, SS, (except CS).
- memory: [BX], [BX+SI+7], 16 bit variable, etc...2

# The Stack

- The stack uses LIFO (Last In First Out) algorithm,
- this means that if we push these values one by one into the stack: 1, 2, 3, 4, 5
- the first value that we will get on pop will be 5, then 4, 3, 2, and only then 1.
- It is very important to do equal number of PUSHs and POPs, otherwise the stack maybe corrupted and it will be impossible to return to operating system.
- As you already know we use RET instruction to return to operating system, so when program starts there is a return address in stack (generally it's 0000h).
- PUSH and POP instruction are especially useful because we don't have too much registers to operate with, so here is a trick:
  - Store original value of the register in stack (using PUSH).
  - Use the register for any purpose.
  - Restore the original value of the register from stack (using POP).

```
ORG 100h
MOV AX, 1234h
PUSH AX ; store value of AX in stack.
MOV AX, 5678h ; modify the AX value.
POP AX ; restore the original value of AX.
RET
END
```

```
ORG 100h
MOV AX, 1212h ; store 1212h in AX.
MOV BX, 3434h ; store 3434h in BX
PUSH AX ; store value of AX in stack.
PUSH BX ; store value of BX in stack.
POP AX ; set AX to original value of BX.
POP BX ; set BX to original value of AX.
RET
END
```

# The Stack

- The stack memory area is set by SS (Stack Segment) register, and SP (Stack Pointer) register. Generally operating system sets values of these registers on program start.
- "PUSH source" instruction does the following:
  - Subtract 2 from SP register.
  - Write the value of source to the address SS:SP.
- "POP destination" instruction does the following:
  - Write the value at the address SS:SP to destination.
  - Add 2 to SP register.
- The current address pointed by SS:SP is called the top of the stack.

# Subroutine

- In computers, a subroutine is a sequence of program instructions that perform a specific task, packaged as a unit.
- This unit can then be used in programs wherever that particular task have to be performed.
- A subroutine is often coded so that it can be started (called) several times and from several places during one execution of the program, including from other subroutines, and then branch back (return) to the next instruction after the call, once the subroutine's task is done.
- It is implemented by using Call and Return instructions.

# Unconditional Call instruction

- CALL address is the format for unconditional call instruction.
- After execution of this instruction program control is transferred to a sub-routine whose starting address is specified in the instruction.
- Value of PC (Program Counter) is transferred to the memory stack and value of SP (Stack Pointer) is decremented by 2.

# Conditional Call instruction

- CC 16-bit address
  - Call at address if cy (carry flag) = 1
- CNC 16-bit address
  - Call at address if cy (carry flag) = 0
- CZ 16-bit address
  - Call at address if ZF (zero flag) = 1
- CNZ 16-bit address
  - Call at address if ZF (zero flag) = 0
- CPE 16-bit address
  - Call at address if PF (parity flag) = 1
- CPO 16-bit address
  - Call at address if PF (parity flag) = 0
- CN 16-bit address
  - Call at address if SF (signed flag) = 1
- CP 16-bit address
  - Call at address if SF (signed flag) = 0

# Unconditional Return instruction

- RET is the instruction used to mark the end of sub-routine.
- It has no parameter. After execution of this instruction program control is transferred back to main program from where it had stopped.
- Value of PC (Program Counter) is retrieved from the memory stack and value of SP (Stack Pointer) is incremented by 2.

# Conditional Return instruction

- RC
  - Return from subroutine if cy (carry flag) = 1
- RNC
  - Return from subroutine if cy (carry flag) = 0
- RZ
  - Return from subroutine if ZF (zero flag) = 1
- RNZ
  - Return from subroutine if ZF (zero flag) = 0
- RPE
  - Return from subroutine if PF (parity flag) = 1
- RPO
  - Return from subroutine if PF (parity flag) = 0
- RN
  - Return from subroutine if SF (signed flag) = 1
- RP
  - Return from subroutine if SF (signed flag) = 0

# Advantages of Subroutine

- Decomposing a complex programming task into simpler steps.
- Reducing duplicate code within a program.
- Enabling reuse of code across multiple programs.
- Improving tractability or makes debugging of a program easy.