Microprocessor Programming

Dr. Tin Ni Ni Kyaw

Ph. D (Kumamoto University, Japan)

Associate Professor

Department of Computer Engineering and Information Technology

Yangon Technological University

Yangon, Myanmar

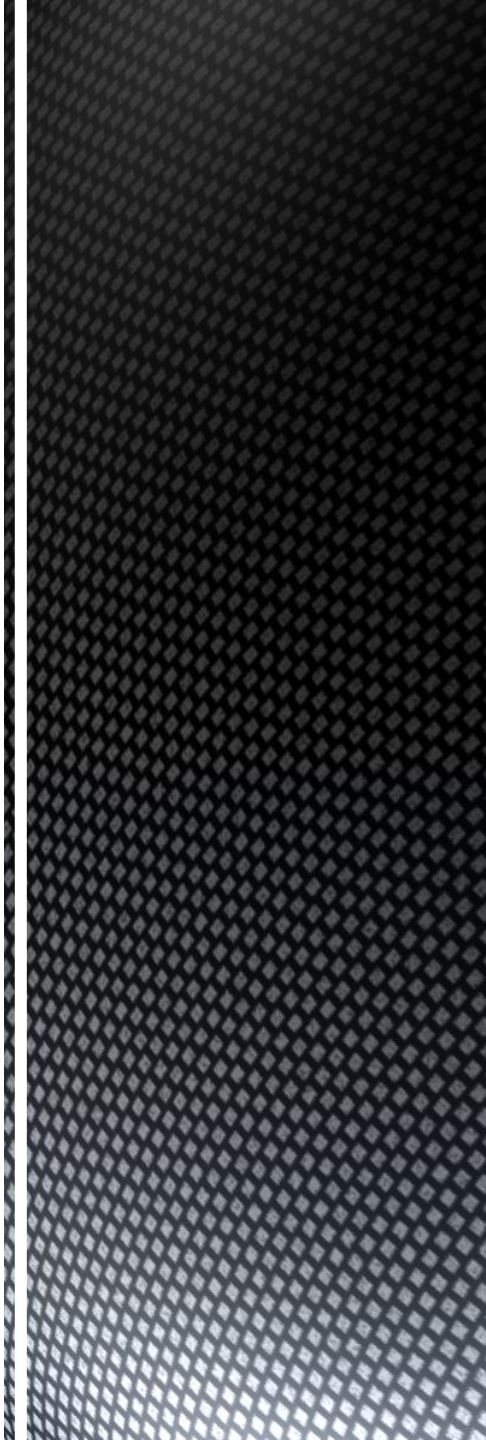
Microprocessor Programming

Lecture 8

Rotate, Stack and Swap

Contents

- Rotate
- Stack
- Swap
- Summary
- Assignments



Rotate Instructions

- With the rotate instructions, the bits from the one end are carried around and inserted into the other.
- The advantage of the rotate instructions is that if the bits are rotated the exact number of times as there are bits in a register or a memory location, the register or the memory location is returned back into its original state.
- As a result, there is no need to save or restore the register or the memory location prior to testing it and the same would apply if one were to rotate the mask instead of the data.

Rotate Instructions (Cont.)

- The formats of the two rotate instructions can be found in Table 1 where ROL means rotate left and ROR means rotate right.

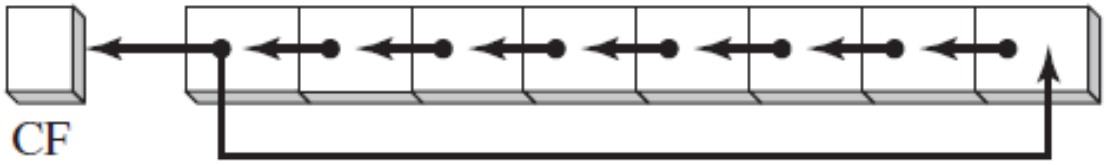
Table 1. Rotate Instructions

Instruction	Meaning
rol reg,imm	Rotate left
rol mem,imm	Rotate left
ror reg,imm	Rotate right
ror mem,imm	Rotate right

Rotate Instructions (Cont.)

ROL Instruction

- The ROL (rotate left) instruction shifts each bit to the left.
- The highest bit is copied into the Carry flag and the lowest bit position.



Rotate Instructions (Cont.)

ROL Instruction

- The instruction format is the same as for SHL (logical shift left).
- The followings are some examples for using ROL instruction to rotate left the content in the register (AL) by one bit.

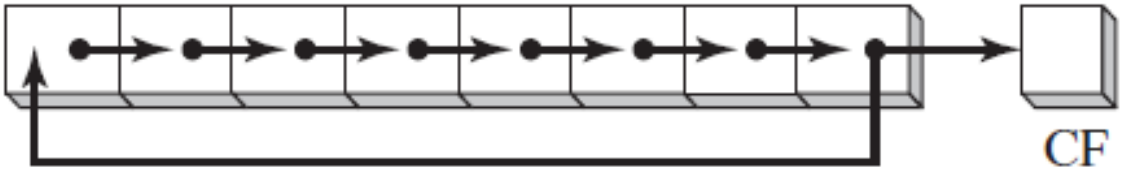
```
mov     al,01010101     ; AL = 01010101
rol     al,1            ; AL = 10101010
```

```
mov     al,10101010     ; AL = 10101010
rol     al,1            ; AL = 01010101
```

Rotate Instructions (Cont.)

ROR Instruction

- The ROR (rotate right) instruction shifts each bit to the right.
- Then, it copies the lowest bit into the Carry flag and the highest bit position.



Rotate Instructions (Cont.)

ROR Instruction

- The instruction format is the same as for SHR (logical shift right).
- The followings are some examples for using ROR instruction to rotate right the content in the register (AL) by one bit.

```
mov     al,01010101     ; AL = 01010101
ror     al,1            ; AL = 10101010
```

```
mov     al,10101010     ; AL = 10101010
ror     al,1            ; AL = 01010101
```

Rotate Instructions (Cont.)

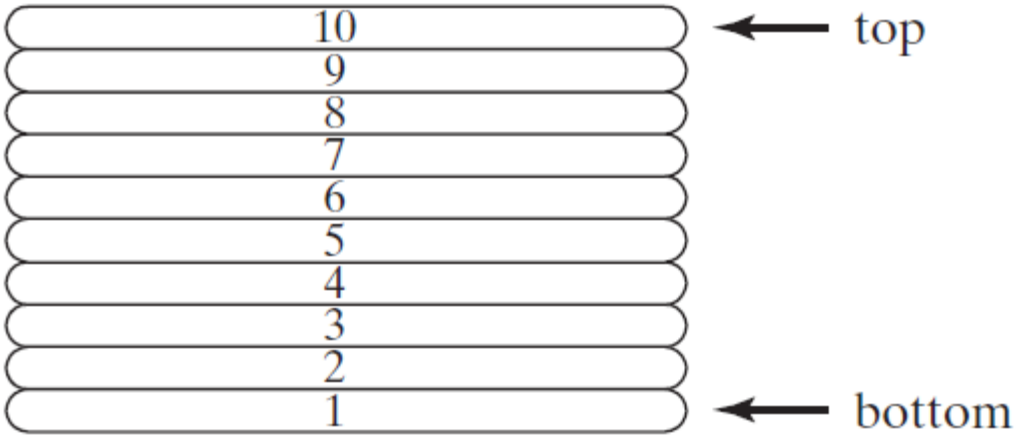
Example

- Testing the 8 bits shown in the previous lecture is now redone below using a rotate instruction instead of a shift instruction:

```
mov count,0                ; initialize count to zero
mov ecx,8                  ; initialize loop counter to zero
.repeat
test al,00000001b          ; test bit position zero
.if !ZERO?                 ; is the bit set?
inc count                  ; yes, count it
.endif
rol al,1                   ; shift al left one bit position
.untilcxz
```

Stack

- A stack is a data structure that follow the same principle as a stack of plates shown in the following figure.
- New values are added to the top of the stack and existing values are removed from the top.



Stack (Cont.)

- In other words, the stack is a LIFO (last in first out) structure where the last item pushed onto the stack is the first one popped off the stack.
- There are a number of useful applications for stacks.
- Some of them are reversing data, matching, number conversions, evaluation of expressions, and implementing recursion.

Stack (Cont.)

- The stack makes a convenient temporary save area for registers when they are used for more than one purpose.
- After they are modified, they can be restored to their original values.
- Note that the last item saved to the stack is the first item that needs to be restored from the stack.

Stack (Cont.)

- The benefit of using the stack is primarily convenience.
- The stack is always available and memory for the stack has already been allocated using the .stack directive.
- So, extra temporary memory locations do not need to be declared and the names of various temporary memory locations do not need to be remembered.

Stack (Cont.)

- Further, it is often useful to hold the original contents of a register or a memory location prior to manipulation of the bit pattern.
- Instead of moving the original bit pattern into a memory location, it can be pushed onto the stack prior to the loop and then restored to its original pattern after the loop.

Stack (Cont.)

- In order to use the stack instructions, make sure to reserve memory space for the stack itself.
- As introduced in Lecture 1, this is accomplished by using the .stack directive which indicates how much memory should be reserved.
- Typically, 100 hexadecimal bytes or 256 decimal bytes is sufficient as shown below:

```
.stack 100h
```

Stack (Cont.)

- To store and retrieve data from the LIFO stack memory, the PUSH and POP instructions are the important instructions.
- To save the data, we can use PUSH instruction and to restore the data, POP instruction can be used.
- Note that only 16-bit or 32-bit registers and memory locations work with the PUSH and POP instructions.

Stack (Cont.)

PUSH

- The instruction PUSH is used to put data on top of the stack.
- PUSH instruction always transfers 16 bits or 32 bits, depending on the register or size of the memory location, of data to the stack.

Stack (Cont.)

POP

- The POP instruction performs the inverse operation of a PUSH instruction.
- The POP instruction removes 16-bit or 32-bit data from the stack and places it into the target register or a memory location.

Stack (Cont.)

- The format of these instructions can be found in Table 2.
- Obviously, it is possible to push an immediate value onto a stack.
- But, it is not possible to pop a value off the stack and put it into an immediate value.

Table 2. PUSH and POP Instructions

PUSH Instruction	POP Instruction
push reg	pop reg
push mem	pop mem
push imm	

Stack (Cont.)

- Here is an example of using the PUSH and POP instructions.

```
push eax                ; save eax register
mov count,0             ; initialize count to zero
mov ecx,8               ; initialize loop counter to
zero
.repeat
test al,00000001b       ; test bit position zero
.if !ZERO?              ; is the bit set?
inc count               ; yes, count it
.endif
shr al,1                ; shift al right one bit position
.untilcxz
pop eax                 ; restore eax register
```

Stack (Cont.)

Example

- Assume that the previous contents of the registers used to evaluate the expression should not be altered, they would need to be saved and restored. How to implement the following arithmetic statement?

```
w = x / y - z;
```

Stack (Cont.)

Example

- Using the stack, the following code segment illustrates how this could be accomplished.

```
push    eax     ; save the original eax
push    edx     ; save the original edx
mov     eax,x   ; eax = x
cdq             ; sign bit extension
idiv    y       ; eax = quotient , edx = remainder
sub     eax,z   ; eax = eax -z
mov     w,eax   ; w = eax
pop     edx     ; restore edx
pop     eax     ; restore eax
```

Swap

- Now, this section will examine how to swap the data values.
- There are three ways for data swapping:

  (1) using registers and mov instruction,

  (2) using the stack and

  (3) using exchange (XCHG) instruction.

Swap (Cont.)

- For example, assume that the two values, num1 and num2, need to be swapped and the typical high-level C code is as follows:

```
temp = num1;
num1 = num2;
num2 = temp;
```

Swap (Cont.)

(1) Using Registers and MOV Instruction

- This could be implemented on a line-by-line basis in assembly language using registers.

```
mov     eax,num1
mov     temp,eax
mov     eax,num2
mov     num1,eax
mov     eax,temp
mov     num2,eax
```

or

```
mov     eax,num1
mov     edx,num2
mov     num1,edx
mov     num2,eax
```

Swap (Cont.)

(2) Using Stack

- Not only a stack is a nice way to save and restore values, it can also be useful in swapping two values.
- An advantage over the MOV instructions is that the stack does not need to use any of the general purpose registers which free them up for other uses.

Swap (Cont.)

(2) Using Stack

- The following code segment swaps the values in num1 and num2 using stack.

```
push    num1
push    num2
pop     num1
pop     num2
```

- Since the purpose of the above code is not to save and restore their contents of num1 and num2 but rather to swap their contents, num1 has to be popped off the stack first to exchange its contents with num2.

Swap (Cont.)

(3) Using XCHG Instruction

- Another method of swapping two values is to use the exchange (XCHG) instruction.
- The XCHG instruction exchanges the contents of a register with the contents of any other register or memory location.
- However, the XCHG instruction cannot exchange memory-to-memory data.

Swap (Cont.)

(3) Using XCHG Instruction

- When swapping two registers, it is faster than the two methods previously presented.
- The format of this instruction is given in Table 3.

Table 3. Exchange Instructions

XCHG Instructions
xchg reg,reg
xchg reg,mem
xchg mem,reg

Swap (Cont.)

(3) Using XCHG Instruction

- This is accomplished by moving one of the two values into a register, swapping the register with the other memory location, and then moving the contents of the register back into the original memory location.

```
mov     eax,num1        ; copy num1 into eax
xchg    eax,num2        ; exchange eax and num2
mov     num1,eax        ; copy eax into num1
```

Summary

- To save and restore the data, the last one pushed onto the stack must be the first one popped off the stack.
- To swap the data, the first one pushed onto the stack must be the first one popped off the stack.
- MOV instructions use more registers but are faster than the PUSH and POP instructions.
- Using the XCHG instruction is a good compromise in terms of register usage and speed.

Microprocessor Programming

Practical Assignments (Instructions)

Assignment 1

- Write a code segment that takes the contents of eax, ebx, ecx, and edx, and puts them in the reverse order of edx, ecx, ebx, and eax using only the PUSH and POP instructions. In other words, eax should contain the contents of edx and vice versa, etc.

Next Lecture

- Procedure
- Implementing Power Function in a Procedure

Thank You