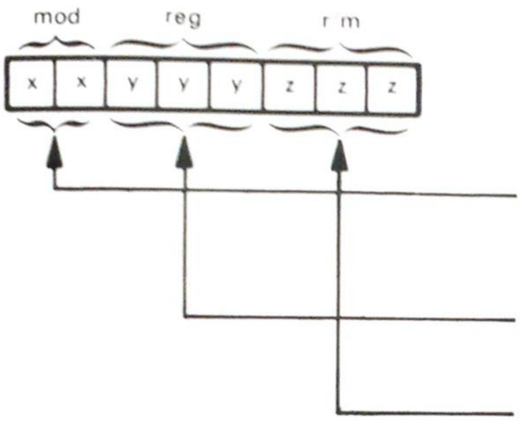
ADDRESSING MODE BYTE

The 8086 obviously offers an extensive selection of addressing modes. The next question is: how are these addressing modes implemented in the object code? The 8086 specifies most data memory addressing modes in an instruction's object code using one byte of object code, known as the addressing mode byte. The addressing mode byte may have one or two additional displacement bytes associated with it. The addressing mode byte is always the second byte of the instruction object code, unless a prefix instruction has been included prior to the initial object code. The addressing mode byte may be illustrated as follows:



xx is two bits that form the mod field. The mod field is used to distinguish between memory and register addressing, and in the case of memory addressing, specifies how many displacement bytes follow the addressing mode byte.

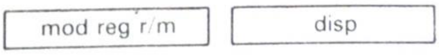
yyy is three bits that form the reg field. The reg field defines which register will be used in the operation. In addition, these three bits may be used to specify instruction.

zzz is three bits that form the r/m field. The r/m field is used in conjunction with the mod field to specify the addressing mode.

mod=

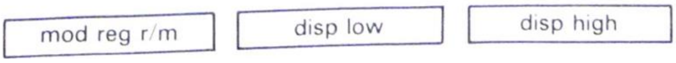
00 Memory addressing mode. r/m specifies the exact addressing option. There are no displacement bytes.

01 Memory addressing mode. r/m specifies the exact addressing option. There is one displacement byte. This displacement byte is viewed as a signed number in the range +127 to −128. When this number is used in the memory address calculation, the number is sign extended to 16 bits. In this case, the addressing mode bytes can be illustrated as follows:



where mod = 01 and disp is the 8-bit signed displacement value.

10 Memory addressing mode. r/m specifies the addressing option. There are two displacement bytes. The first displacement byte is the low-order eight bits of the displacement. The second displacement byte is the high-order eight bits of the displacement. When this number is used in the memory address calculation, the number is treated as an unsigned 16-bit number. In this case, the addressing mode bytes can be illustrated as follows:



where mod = 10, disp low is the low-order eight bits of the displacement, and disp high is the high-order eight bits of the displacement.

11 register addressing mode. r/m specifies a register. Used in conjunction with the w bit to determine if an 8- or 16-bit register is selected.

reg reg is used in conjunction with another bit, the w bit, in the selection of the register to be used in the operation. The w bit, which is part of the instruction op-code, selects whether an 8- or 16-bit operation is performed.

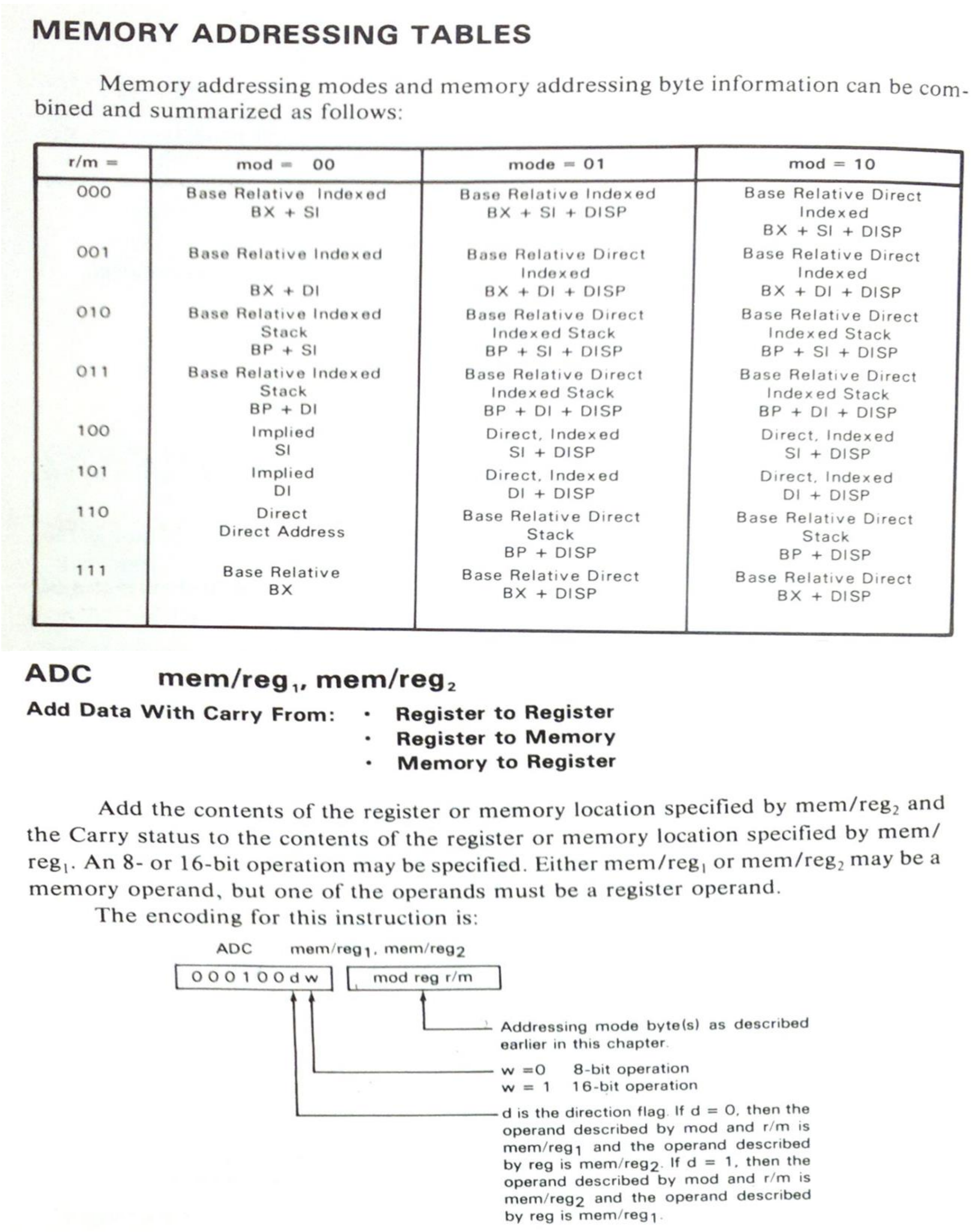
reg	w = 0	w = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

r/m r/m specifies the addressing mode in conjunction with mod, as follows:

r/m	mod - 00	mod - 01	mod - 10	mod - 11	
				w = 0	w = 1
000	BX + SI	BX + SI + DISP	BX + SI + DISP	AL	AX
001	BX + DI	BX + DI + DISP	BX + DI + DISP	CL	CX
010	BP + SI	BP + SI + DISP	BP + SI + DISP	DL	DX
011	BP + DI	BP + DI + DISP	BP + DI + DISP	BL	BX
100	SI	SI + DISP	SI + DISP	AH	SP
101	DI	DI + DISP	DI + DISP	CH	BP
110	Direct Address	BP + DISP	BP + DISP	DH	SI
111	BX	BX + DISP	BX + DISP	BH	DI

This table is self-explanatory, with the exception of Direct Address. When mod is 00 and r/m is 110, the offset address is taken directly from the two bytes that follow the addressing mode byte. This can be illustrated as follows:

mod reg r m	addr-low	addr-high



MEMORY ADDRESSING TABLES

Memory addressing modes and memory addressing byte information can be combined and summarized as follows:

r/m =	mod = 00	mode = 01	mod = 10
000	Base Relative Indexed BX + SI	Base Relative Indexed BX + SI + DISP	Base Relative Direct Indexed BX + SI + DISP
001	Base Relative Indexed BX + DI	Base Relative Direct Indexed BX + DI + DISP	Base Relative Direct Indexed BX + DI + DISP
010	Base Relative Indexed Stack BP + SI	Base Relative Direct Indexed Stack BP + SI + DISP	Base Relative Direct Indexed Stack BP + SI + DISP
011	Base Relative Indexed Stack BP + DI	Base Relative Direct Indexed Stack BP + DI + DISP	Base Relative Direct Indexed Stack BP + DI + DISP
100	Implied SI	Direct, Indexed SI + DISP	Direct, Indexed SI + DISP
101	Implied DI	Direct, Indexed DI + DISP	Direct, Indexed DI + DISP
110	Direct Direct Address	Base Relative Direct Stack BP + DISP	Base Relative Direct Stack BP + DISP
111	Base Relative BX	Base Relative Direct BX + DISP	Base Relative Direct BX + DISP

ADC mem/reg$_1$, mem/reg$_2$

Add Data With Carry From:
- **Register to Register**
- **Register to Memory**
- **Memory to Register**

Add the contents of the register or memory location specified by mem/reg$_2$ and the Carry status to the contents of the register or memory location specified by mem/reg$_1$. An 8- or 16-bit operation may be specified. Either mem/reg$_1$ or mem/reg$_2$ may be a memory operand, but one of the operands must be a register operand.

The encoding for this instruction is:

ADC mem/reg$_1$, mem/reg$_2$

000100dw | mod reg r/m

Addressing mode byte(s) as described earlier in this chapter.

w = 0 8-bit operation
w = 1 16-bit operation

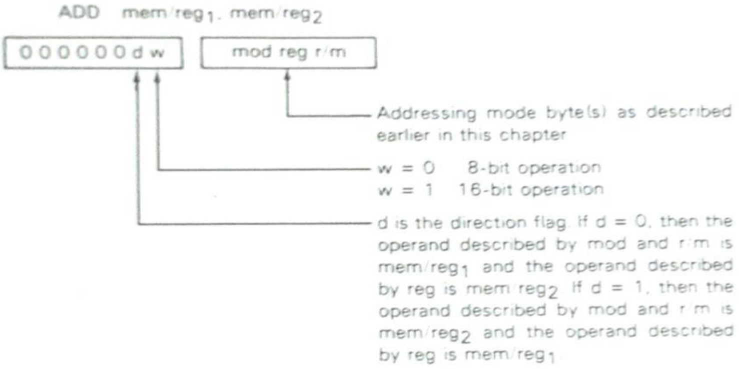
d is the direction flag. If d = 0, then the operand described by mod and r/m is mem/reg$_1$ and the operand described by reg is mem/reg$_2$. If d = 1, then the operand described by mod and r/m is mem/reg$_2$ and the operand described by reg is mem/reg$_1$.

ADD mem/reg$_1$, mem/reg$_2$

Add: 1. Register to Register
2. Register to Memory
3. Memory to Register

Add the contents of the register or memory location specified by mem/reg$_2$ to the contents of the register or memory location specified by mem/reg$_1$. An 8- or 16-bit operation may be specified. Either mem/reg$_1$ or mem/reg$_2$ may be a memory operand, but one of the operands must be a register operand.

The encoding for this instruction is:



Addressing mode byte(s) as described earlier in this chapter

w = 0 8-bit operation
w = 1 16-bit operation

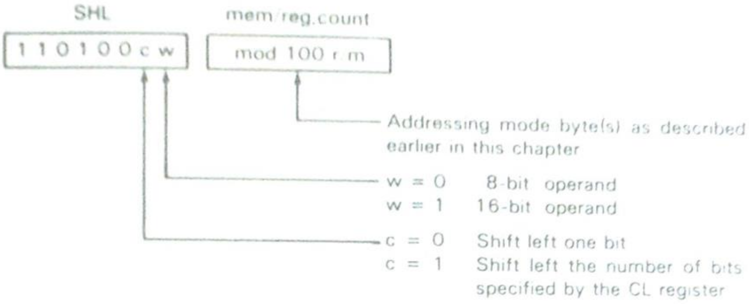
d is the direction flag. If d = 0, then the operand described by mod and r/m is mem/reg$_1$ and the operand described by reg is mem/reg$_2$. If d = 1, then the operand described by mod and r/m is mem/reg$_2$ and the operand described by reg is mem/reg$_1$.

SHL mem/reg,count
SAL mem/reg,count

Shift Register or Memory Location Left

Shift the contents of the specified register or memory location left by the specified number of bits. The number of bits to shift, represented by the variable count, is either one or the number contained in the CL register. This is a logical left shift.

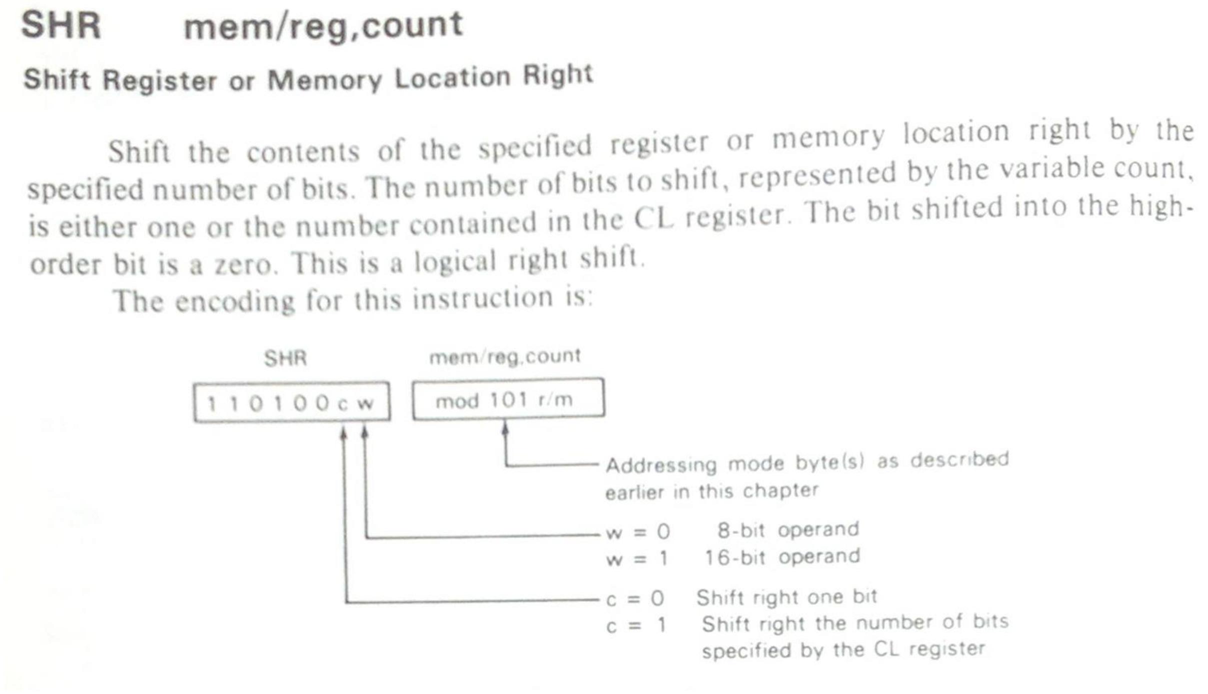
The encoding for this instruction is:



Addressing mode byte(s) as described earlier in this chapter

w = 0 8-bit operand
w = 1 16-bit operand

c = 0 Shift left one bit
c = 1 Shift left the number of bits specified by the CL register



SHR mem/reg,count

Shift Register or Memory Location Right

Shift the contents of the specified register or memory location right by the specified number of bits. The number of bits to shift, represented by the variable count, is either one or the number contained in the CL register. The bit shifted into the high-order bit is a zero. This is a logical right shift.

The encoding for this instruction is:

SHR: 1 1 0 1 0 0 c w

mem/reg,count: mod 101 r/m

Addressing mode byte(s) as described earlier in this chapter

w = 0 8-bit operand
w = 1 16-bit operand

c = 0 Shift right one bit
c = 1 Shift right the number of bits specified by the CL register

Adressing Mode Byte Örnekleri

Opcode (*binary*) Add. ModeByte (*binary*) Machine Code(*Hexadecimal*)

```
ADC AX,BX              ;
ADC [BX],AX            ;
SHR AX,1               ;
SHR AX,CL              ;
SHR [BX],1             ;
SHR WORD PTR[BX],1     ;
```

Machine Code(*Hexadecimal*) Opcode (*binary*) Add. ModeByte (*binary*) Assembly Language Instruction

03EAh →

0001h →

D1E0h →

D32Ch →