Intel machine code

CSC 236

Machine code

- Microprocessor
  - Executes machine instructions
  - Binary (hex) format
- Assembler source
  - Human-readable
  - Mnemonic representation of machine instructions
  - (And some more: data declaration, labels, directives)
- Assembler & linker
  - Create machine code

Two microprocessor architectures

CISC

- Complex instruction set computer
- 1960-1990
- Microcode
- Lots of function
- Irregular instructions
  - Size
  - Time
- Direct memory access

RISC

- Reduced instruction set computer
- 1990+
- Hardwired
- Few functions
- Fixed size, time
- Pipelining
- Superscalar
- Load-store architecture
  - No direct memory access

X86 is CISC

Basic rules

Details

Instruction length

- 80386 -- pentium
  - 32-bit addressing
  - Machine instructions can be up to 15 bytes long
- Core
  - 64-bit addressing
  - Machine instructions can be up to 23 bytes long
- 8086
  - 16-bit addressing
  - Machine instructions can be up to 6 bytes long

8086 instruction

code	[addr]	[disp]		[data]	
opcode d w 6 1 1	mod reg r/m 2 3 3	disp low	disp high	data low	data high

- Up to 6 bytes (minimum of 1)
- Four parts
  - Code
  - Addressing
  - Displacement — for memory references
  - Data — for immediate values

8086 instruction

code	[addr]	[disp]		[data]	
opcode d w 6 1 1	mod reg r/m 2 3 3	disp low	disp high	data low	data high

- Code
  - Opcode — defines instruction (6 bits = 64 unique opcodes)
  - d — data direction (eg, reg from/to mem)
  - w — size: 1 ⇒ word; 0 ⇒ byte
- This one generic form of the code byte

8086 instruction

code	[addr]	[disp]		[data]	
opcode d w 6 1 1	mod reg r/m 2 3 3	disp low	disp high	data low	data high

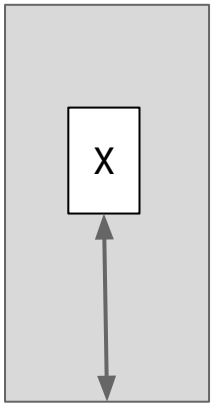
- addr byte
  - Two modes
  - Specifies operands
  - Extends opcode information
  - Optional
    - Some operands are implicit

- 3 fields
  - mod Mode
  - reg Register
  - r/m Register or memory
- Tells
  - Operands in reg or memory
  - Extends opcode
  - Identifies a register

8086 instruction

code	[addr]	[disp]		[data]	
opcode d w 6 1 1	mod reg r/m 2 3 3	disp low	disp high	data low	data high

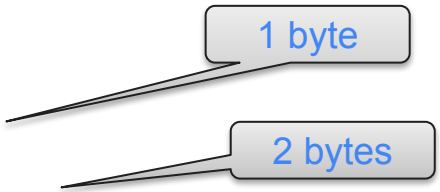
- disp — displacement
  - Offset into memory
  - Always two bytes if present
  - Reverse byte order
  - Eg, `inc [X]`



8086 instruction

code	[addr]	[disp]		[data]	
opcode d w 6 1 1	mod reg r/m 2 3 3	disp low	disp high	data low	data high

- data — immediate data
  - Immediate value
  - Byte or word (1 or 2 bytes)
  - Eg, `add al,5`
  - Eg, `add ax,3A7Bh`



Summary of disp and data fields

- [disp] if instruction has memory reference
  - `mov ax,[var]`

- [data] if instruction has immediate data
  - `mov ax,1000`

  - `mov al,100`

- Both if instruction has memory reference and immediate data
  - `mov [var],1000`

Largest instruction -- 6 bytes

Conversion

- Convert
  - ASM to machine
  - Machine to ASM
- Three tables
  - Table 1 — general information
  - Table 2 — ASM to machine
  - Table 3 — machine to ASM

Example 1

- `mov ax, [var]`
  - Assembler picks location for var
  - We don’t know it (when we write)
  - Suppose var is at offset 12_{10}
  - [disp] = 0C 00

Example 1

- `mov ax, [var]`
  - Look in Table 2
  - Several choices

mov						
mov reg/mem to reg/mem	1000 10dw	mod reg r/m	disp-lo	disp-hi		
mov immed to mem	1100 011w	mod 000 r/m	disp-lo	disp-hi	data-lo	data-hi
mov immed to reg	1011 w reg	data-lo	data-hi			
mov mem to accumulator	1010 000w	disp-lo	disp-hi			
mov accumulator to mem	1010 001w	disp-lo	disp-hi			
mov reg/mem to seg reg	1000 1110	mod 0 sr r/m	disp-lo	disp-hi		
mov seg reg to reg/mem	1000 1100	mod 0 sr r/m	disp-lo	disp-hi		

Example 1

- `mov ax, [var]`
  - Look in Table 2
  - Several choices

mov						
mov reg/mem to reg/mem ✔	1000 10dw	mod reg r/m	disp-lo	disp-hi		
mov immed to mem	1100 011w	mod 000 r/m	disp-lo	disp-hi	data-lo	data-hi
mov immed to reg	1011 w reg	data-lo	data-hi			
mov mem to accumulator	1010 000w	disp-lo	disp-hi			
mov accumulator to mem	1010 001w	disp-lo	disp-hi			
mov reg/mem to seg reg	1000 1110	mod 0 sr r/m	disp-lo	disp-hi		
mov seg reg to reg/mem	1000 1100	mod 0 sr r/m	disp-lo	disp-hi		

Example 1

- `mov ax, [var]`
  - Look in Table 2
  - Several choices

mov						
mov reg/mem to reg/mem ✔	1000 10dw	mod reg r/m	disp-lo	disp-hi		
mov immed to mem ✘	1100 011w	mod 000 r/m	disp-lo	disp-hi	data-lo	data-hi
mov immed to reg	1011 w reg	data-lo	data-hi			
mov mem to accumulator	1010 000w	disp-lo	disp-hi			
mov accumulator to mem	1010 001w	disp-lo	disp-hi			
mov reg/mem to seg reg	1000 1110	mod 0 sr r/m	disp-lo	disp-hi		
mov seg reg to reg/mem	1000 1100	mod 0 sr r/m	disp-lo	disp-hi		

Example 1

- `mov ax, [var]`
  - Look in Table 2
  - Several choices

mov						
mov reg/mem to reg/mem ✔	1000 10dw	mod reg r/m	disp-lo	disp-hi		
mov immed to mem ✘	1100 011w	mod 000 r/m	disp-lo	disp-hi	data-lo	data-hi
mov immed to reg ✘	1011 w reg	data-lo	data-hi			
mov mem to accumulator	1010 000w	disp-lo	disp-hi			
mov accumulator to mem	1010 001w	disp-lo	disp-hi			
mov reg/mem to seg reg	1000 1110	mod 0 sr r/m	disp-lo	disp-hi		
mov seg reg to reg/mem	1000 1100	mod 0 sr r/m	disp-lo	disp-hi		

Example 1

- `mov ax, [var]`
  - Look in Table 2
  - Several choices

mov						
mov reg/mem to reg/mem ✔	1000 10dw	mod reg r/m	disp-lo	disp-hi		
mov immed to mem ✘	1100 011w	mod 000 r/m	disp-lo	disp-hi	data-lo	data-hi
mov immed to reg ✘	1011 w reg	data-lo	data-hi			
mov mem to accumulator ✔	1010 000w	disp-lo	disp-hi			
mov accumulator to mem	1010 001w	disp-lo	disp-hi			
mov reg/mem to seg reg	1000 1110	mod 0 sr r/m	disp-lo	disp-hi		
mov seg reg to reg/mem	1000 1100	mod 0 sr r/m	disp-lo	disp-hi		

Example 1

- `mov ax, [var]`
  - Look in Table 2
  - Several choices

mov

mov reg/mem to reg/mem ✔	1000 10dw	mod reg r/m	disp-lo	disp-hi		
mov immed to mem ✘	1100 011w	mod 000 r/m	disp-lo	disp-hi	data-lo	data-hi
mov immed to reg ✘	1011 w reg	data-lo	data-hi			
mov mem to accumulator ✔	1010 000w	disp-lo	disp-hi			
mov accumulator to mem ✘	1010 001w	disp-lo	disp-hi			
mov reg/mem to seg reg	1000 1110	mod 0 sr r/m	disp-lo	disp-hi		
mov seg reg to reg/mem	1000 1100	mod 0 sr r/m	disp-lo	disp-hi		

Example 1

- `mov ax, [var]`
  - Look in Table 2
  - Several choices

mov

mov reg/mem to reg/mem ✔	1000 10dw	mod reg r/m	disp-lo	disp-hi		
mov immed to mem ✘	1100 011w	mod 000 r/m	disp-lo	disp-hi	data-lo	data-hi
mov immed to reg ✘	1011 w reg	data-lo	data-hi			
mov mem to accumulator ✔	1010 000w	disp-lo	disp-hi			
mov accumulator to mem ✘	1010 001w	disp-lo	disp-hi			
mov reg/mem to seg reg ✘	1000 1110	mod 0 sr r/m	disp-lo	disp-hi		
mov seg reg to reg/mem	1000 1100	mod 0 sr r/m	disp-lo	disp-hi		

Example 1

- `mov ax, [var]`
  - Look in Table 2
  - Several choices

mov

mov reg/mem to reg/mem ✔	1000 10dw	mod reg r/m	disp-lo	disp-hi		
mov immed to mem ✘	1100 011w	mod 000 r/m	disp-lo	disp-hi	data-lo	data-hi
mov immed to reg ✘	1011 w reg	data-lo	data-hi			
mov mem to accumulator ✔	1010 000w	disp-lo	disp-hi			
mov accumulator to mem ✘	1010 001w	disp-lo	disp-hi			
mov reg/mem to seg reg ✘	1000 1110	mod 0 sr r/m	disp-lo	disp-hi		
mov seg reg to reg/mem ✘	1000 1100	mod 0 sr r/m	disp-lo	disp-hi		

Example 1

- `mov ax, [var]`
  - Look in Table 2
  - Several choices

The accumulator is used more often than other registers.
Eg, CBW, CWD, MUL, DIV ...

mov

mov reg/mem to reg/mem ✔	1000 10dw	mod reg r/m	disp-lo	disp-hi		
mov immed to mem ✘	1100 011w	mod 000 r/m	disp-lo	disp-hi	data-lo	data-hi
mov immed to reg ✘	1011 w reg	data-lo	data-hi			
mov mem to accumulator ✔	1010 000w	disp-lo	disp-hi			
mov accumulator to mem ✘	1010 001w	disp-lo	disp-hi			
mov reg/mem to seg reg ✘	1000 1110	mod 0 sr r/m	disp-lo	disp-hi		
mov seg reg to reg/mem ✘	1000 1100	mod 0 sr r/m	disp-lo	disp-hi		

Example 1

- `mov ax, [var]`
  - Look in Table 2
  - Several choices
1010 000w	disp lo	disp hi

mov

mov reg/mem to reg/mem ✔	1000 10dw	mod reg r/m	disp-lo	disp-hi		
mov immed to mem ✘	1100 011w	mod 000 r/m	disp-lo	disp-hi	data-lo	data-hi
mov immed to reg ✘	1011 w reg	data-lo	data-hi			
mov mem to accumulator ✔	1010 000w	disp-lo	disp-hi			
mov accumulator to mem ✘	1010 001w	disp-lo	disp-hi			
mov reg/mem to seg reg ✘	1000 1110	mod 0 sr r/m	disp-lo	disp-hi		
mov seg reg to reg/mem ✘	1000 1100	mod 0 sr r/m	disp-lo	disp-hi		

Example 1

- `mov ax, [var]`
  - Look in Table 2
  - Several choices
  - 

- What is w?
  - Size: 1 = word; 0 = byte
  - ax is word

Example 1

- `mov ax, [var]`
  - Look in Table 2
  - Several choices
1010 000w	disp lo	disp hi
1010 0001	0C	00
  - A1 0C 00

Example 1 a & b

- mov al,[var]
- mov ah,[var]

Example 1 a & b

- mov al,[var]

1010 000w	disp lo	disp hi
1010 0000	0C	00

A0 0C 00

- Accumulator
  - Word = ax
  - Byte = al (not ah)

- mov ah,[var]

mov

mov reg/mem to reg/mem ✔	1000 10dw	mod reg r/m	disp-lo	disp-hi		
mov immed to mem	1100 011w	mod 000 r/m	disp-lo	disp-hi	data-lo	data-hi
mov immed to reg	1011 w reg	data-lo	data-hi			
mov mem to accumulator ✔	1010 000w	disp-lo	disp-hi			
mov accumulator to mem	1010 001w	disp-lo	disp-hi			
mov reg/mem to seg reg	1000 1110	mod 0 sr r/m	disp-lo	disp-hi		
mov seg reg to reg/mem	1000 1100	mod 0 sr r/m	disp-lo	disp-hi		

Example 1 a

- mov al,[var]

1010 000w	disp lo	disp hi
1010 0000	0C	00

A0 0C 00

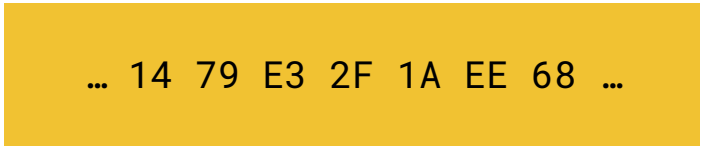
- Accumulator
  - Word = ax
  - Byte = al (not ah)

- mov ah,[var]

Later

Decoding

- Main memory
- Are these bytes
  - Data or
  - Instructions?



Decoding

- Main memory

```
… 14 79 E3 2F 1A EE 68 …
```

- Are these bytes
  - Data or
  - Instructions?
  - Cannot determine
- Code segment memory

Decoding

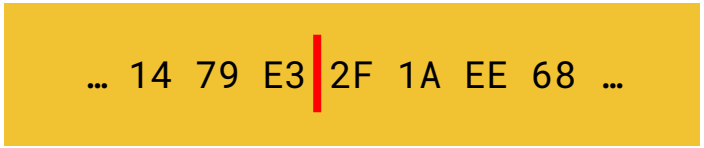
- Main memory

```
… 14 79 E3 2F 1A EE 68 …
```

- Are these bytes
  - Data or
  - Instructions?
  - Cannot determine
- Code segment memory
  - Still cannot tell
  - Overrides allow code or data in any segment

Decoding

- Main memory



- Need to know code beginning
- Can you decode backwards?
  - Not in general
- Can you decode forwards?
  - Better be able to!

Example 2

- Convert this machine code
  - `80 06 0B 00 14 …`

Table-3 8086 Instruction Decoding (Machine to Assembler

push seg reg	000 sr 110			
pop seg reg	000 sr 111			
add reg/mem to reg/mem	0000 00dw	mod reg r/m	disp-lo	disp-hi
add immed to accumulator	0000 010w	data-lo	data-hi	
or reg/mem to reg/mem	0000 10dw	mod reg r/m	disp-lo	disp-hi
or immed to accumulator	0000 110w	data-lo	data-hi	
adc reg/mem to reg/mem	0001 00dw	mod reg r/m	disp-lo	disp-hi
adc immed to accumulator	0001 010w	data-lo	data-hi	

Example 2

Look at code byte = `1000 0000`

- Convert this machine code
  - `80 06 0B 00 14 …`

Table-3 8086 Instruction Decoding (Machine to Assembler

push seg reg	000 sr 110			
pop seg reg	000 sr 111			
add reg/mem to reg/mem	0000 00dw	mod reg r/m	disp-lo	disp-hi
add immed to accumulator	0000 010w	data-lo	data-hi	
or reg/mem to reg/mem	0000 10dw	mod reg r/m	disp-lo	disp-hi
or immed to accumulator	0000 110w	data-lo	data-hi	
adc reg/mem to reg/mem	0001 00dw	mod reg r/m	disp-lo	disp-hi
adc immed to accumulator	0001 010w	data-lo	data-hi	

Example 2

Look at code byte = `1000 0000`

- Convert this machine code
  - `80 06 0B 00 14 …`
- Table 3

add immed to reg/mem	1000 000w	mod 000 r/m	disp-lo	disp-hi	data-lo	data-hi
or immediate to reg/mem	1000 000w	mod 001 r/m	disp-lo	disp-hi	data-lo	data-hi
adc immed to reg/mem	1000 000w	mod 010 r/m	disp-lo	disp-hi	data-lo	data-hi
sbb immed from reg/mem	1000 000w	mod 011 r/m	disp-lo	disp-hi	data-lo	data-hi
and immed to reg/mem	1000 000w	mod 100 r/m	disp-lo	disp-hi	data-lo	data-hi
sub immed from reg/mem	1000 000w	mod 101 r/m	disp-lo	disp-hi	data-lo	data-hi
xor immed to reg/mem	1000 000w	mod 110 r/m	disp-lo	disp-hi	data-lo	data-hi
cmp reg/mem to immed	1000 000w	mod 111 r/m	disp-lo	disp-hi	data-lo	data-hi

Example 2

- Convert this machine code
  - `80 06 0B 00 14 …`
- Table 3

Look at code byte = `1000 0000`

w = 0 ⇒ size is byte

add immed to reg/mem	1000 000w	mod 000 r/m	disp-lo	disp-hi	data-lo	data-hi
or immediate to reg/mem	1000 000w	mod 001 r/m	disp-lo	disp-hi	data-lo	data-hi
adc immed to reg/mem	1000 000w	mod 010 r/m	disp-lo	disp-hi	data-lo	data-hi
sbb immed from reg/mem	1000 000w	mod 011 r/m	disp-lo	disp-hi	data-lo	data-hi
and immed to reg/mem	1000 000w	mod 100 r/m	disp-lo	disp-hi	data-lo	data-hi
sub immed from reg/mem	1000 000w	mod 101 r/m	disp-lo	disp-hi	data-lo	data-hi
xor immed to reg/mem	1000 000w	mod 110 r/m	disp-lo	disp-hi	data-lo	data-hi
cmp reg/mem to immed	1000 000w	mod 111 r/m	disp-lo	disp-hi	data-lo	data-hi

Example 2

Look at code byte = `1000 0000`

- Convert this machine code
  - `80 06 0B 00 14 …`
- Table 3

Now look at addr byte
`0000 0110 = 00 000 110`

add immed to reg/mem	1000 000w	mod 000 r/m	disp-lo	disp-hi	data-lo	data-hi
or immediate to reg/mem	1000 000w	mod 001 r/m	disp-lo	disp-hi	data-lo	data-hi
adc immed to reg/mem	1000 000w	mod 010 r/m	disp-lo	disp-hi	data-lo	data-hi
sbb immed from reg/mem	1000 000w	mod 011 r/m	disp-lo	disp-hi	data-lo	data-hi
and immed to reg/mem	1000 000w	mod 100 r/m	disp-lo	disp-hi	data-lo	data-hi
sub immed from reg/mem	1000 000w	mod 101 r/m	disp-lo	disp-hi	data-lo	data-hi
xor immed to reg/mem	1000 000w	mod 110 r/m	disp-lo	disp-hi	data-lo	data-hi
cmp reg/mem to immed	1000 000w	mod 111 r/m	disp-lo	disp-hi	data-lo	data-hi

Example 2

Look at code byte = `1000 0000`

- Convert this machine code
  - `80 06 0D 00 14 …`
- Table 3

Now look at addr byte
`0000 0110 = 00 000 110`

add immed to reg/mem ✔	1000 000w	mod	000	r/m	disp-lo	disp-hi	data-lo	data-hi
or immediate to reg/mem	1000 000w	mod	001	r/m	disp-lo	disp-hi	data-lo	data-hi
adc immed to reg/mem	1000 000w	mod	010	r/m	disp-lo	disp-hi	data-lo	data-hi
sbb immed from reg/mem	1000 000w	mod	011	r/m	disp-lo	disp-hi	data-lo	data-hi
and immed to reg/mem	1000 000w	mod	100	r/m	disp-lo	disp-hi	data-lo	data-hi
sub immed from reg/mem	1000 000w	mod	101	r/m	disp-lo	disp-hi	data-lo	data-hi
xor immed to reg/mem	1000 000w	mod	110	r/m	disp-lo	disp-hi	data-lo	data-hi
cmp reg/mem to immed	1000 000w	mod	111	r/m	disp-lo	disp-hi	data-lo	data-hi

Example 2

Look at code byte = `1000 0000`

- Convert this machine code
  - `80 06 0D 00 14 …`
- Table 3
  - add immed to reg/mem

Now look at addr byte
`0000 0110 = 00 000 110`

add immed to reg/mem ✔	1000 000w	mod	000	r/m	disp-lo	disp-hi	data-lo	data-hi
or immediate to reg/mem	1000 000w	mod	001	r/m	disp-lo	disp-hi	data-lo	data-hi
adc immed to reg/mem	1000 000w	mod	010	r/m	disp-lo	disp-hi	data-lo	data-hi
sbb immed from reg/mem	1000 000w	mod	011	r/m	disp-lo	disp-hi	data-lo	data-hi
and immed to reg/mem	1000 000w	mod	100	r/m	disp-lo	disp-hi	data-lo	data-hi
sub immed from reg/mem	1000 000w	mod	101	r/m	disp-lo	disp-hi	data-lo	data-hi
xor immed to reg/mem	1000 000w	mod	110	r/m	disp-lo	disp-hi	data-lo	data-hi
cmp reg/mem to immed	1000 000w	mod	111	r/m	disp-lo	disp-hi	data-lo	data-hi

Example 2

- Convert this machine code
  - `80 06 0B 00 14 …`
- Table 3
  - add immed to reg/mem

Now look at addr byte

`0000 0110 = 00 000 110`

Table 1

OPERAND(S)	MOD	REG	R/M	DISP	IMM
1. single register	11	special code	register		
2. single memory	00	special code	110	mem addr	
3. immediate to register	11	special code	register		data
4. immediate to memory	00	special code	110	mem addr	data
5. register and memory	00	register	110	mem addr	
6. register and register	11	register_1	register_2		

Example 2

- Convert this machine code
  - `80 06 0B 00 14 …`
- Table 3
  - add immed to reg/mem

Now look at addr byte

`0000 0110 = 00 000 110`

	OPERAND(S)	MOD	REG	R/M	DISP	IMM
✘	1. single register	11	special code	register		
	2. single memory	00	special code	110	mem addr	
✘	3. immediate to register	11	special code	register		data
	4. immediate to memory	00	special code	110	mem addr	data
	5. register and memory	00	register	110	mem addr	
✘	6. register and register	11	register_1	register_2		

Example 2

- Convert this machine code
  - `80 06 0B 00 14 …`
- Table 3
  - add immed to reg/mem

Now look at addr byte

`0000 0110 = 00 000 110`

	OPERAND(S)	MOD	REG	R/M	DISP	IMM
✘	1. single register	11	special code	register		
✘	2. single memory	00	special code	110	mem addr	
✘	3. immediate to register	11	special code	register		data
	4. immediate to memory	00	special code	110	mem addr	data
	5. register and memory	00	register	110	mem addr	
✘	6. register and register	11	register_1	register_2		

Example 2

- Convert this machine code
  - `80 06 0B 00 14 …`
- Table 3
  - add immed to reg/mem

Now look at addr byte

`0000 0110 = 00 000 110`

	OPERAND(S)	MOD	REG	R/M	DISP	IMM
✘	1. single register	11	special code	register		
✘	2. single memory	00	special code	110	mem addr	
✘	3. immediate to register	11	special code	register		data
✔	4. immediate to memory	00	special code	110	mem addr	data
	5. register and memory	00	register	110	mem addr	
✘	6. register and register	11	register_1	register_2		

Example 2

- Convert this machine code
  - `80 06 0B 00 14 …`
- Table 3
  - add immed to reg/mem

Now look at addr byte

`0000 0110 = 00 000 110`

	OPERAND(S)	MOD	REG	R/M	DISP	IMM
✗	1. single register	11	special code	register		
✗	2. single memory	00	special code	110	mem addr	
✗	3. immediate to register	11	special code	register		data
✓	4. immediate to memory	00	special code	110	mem addr	data
✗	5. register and memory	00	register	110	mem addr	
✗	6. register and register	11	register_1	register_2		

Example 2

- Convert this machine code
  - `80 06 0B 00 14 …`
- Table 3
  - add immed to ~~reg~~/mem

Now look at addr byte

`0000 0110 = 00 000 110`

add immed to reg/mem	1000 000w	mod 000 r/m	disp-lo	disp-hi	data-lo	data-hi

OPERAND(S)	MOD	REG	R/M	DISP	IMM
1. single register	11	special code	register		
2. single memory	00	special code	110	mem addr	
3. immediate to register	11	special code	register		data
4. immediate to memory	00	special code	110	mem addr	data
5. register and memory	00	register	110	mem addr	
6. register and register	11	register_1	register_2		

Example 2

- Convert this machine code
  - 80 06 0B 00 14 …
- Table 3
  - add immed to ~~reg~~/mem

Now look at addr byte

0000 0110 = 00 000 110

OPERAND(S)	MOD	REG	R/M	DISP	IMM
1. single register	11	special code	register		
2. single memory	00	special code	110	mem addr	
3. immediate to register	11	special code	register		data
4. immediate to memory	00	special code	110	mem addr	data
5. register and memory	00	register	110	mem addr	
6. register and register	11	register_1	register_2		

Example 2

- Convert this machine code
  - `80 06 0B 00 14 …`
- Table 3
  - add immed to ~~reg~~/mem
  - `add [var],14h  ;var is at offset 000B = 11`$_{10}$

Now look at addr byte

`0000 0110 = 00 000 110`

	OPERAND(S)	MOD	REG	R/M	DISP	IMM
✗	1. single register	11	special code	register		
✗	2. single memory	00	special code	110	mem addr	
✗	3. immediate to register	11	special code	register		data
✓	4. immediate to memory	00	special code	110	mem addr	data
✗	5. register and memory	00	register	110	mem addr	
✗	6. register and register	11	register_1	register_2		

Example 3

- Convert
  - `mov dx,[var]`
  - var at offset 12_{10} into data segment
- This is similar to example 1
  - `mov ax,[var]`
  - ax is *accumulator* has many shortcuts

Example 3

- `mov dx, [var]`
  - Look in Table 2
  - Several choices

mov						
mov reg/mem to reg/mem	1000 10dw	mod reg r/m	disp-lo	disp-hi		
mov immed to mem	1100 011w	mod 000 r/m	disp-lo	disp-hi	data-lo	data-hi
mov immed to reg	1011 w reg	data-lo	data-hi			
mov mem to accumulator	1010 000w	disp-lo	disp-hi			
mov accumulator to mem	1010 001w	disp-lo	disp-hi			
mov reg/mem to seg reg	1000 1110	mod 0 sr r/m	disp-lo	disp-hi		
mov seg reg to reg/mem	1000 1100	mod 0 sr r/m	disp-lo	disp-hi		

Example 3

- `mov dx, [var]`
  - Look in Table 2
  - Several choices

mov						
mov reg/mem to reg/mem ✔	1000 10dw	mod reg r/m	disp-lo	disp-hi		
mov immed to mem	1100 011w	mod 000 r/m	disp-lo	disp-hi	data-lo	data-hi
mov immed to reg	1011 w reg	data-lo	data-hi			
mov mem to accumulator	1010 000w	disp-lo	disp-hi			
mov accumulator to mem	1010 001w	disp-lo	disp-hi			
mov reg/mem to seg reg	1000 1110	mod 0 sr r/m	disp-lo	disp-hi		
mov seg reg to reg/mem	1000 1100	mod 0 sr r/m	disp-lo	disp-hi		

Example 3

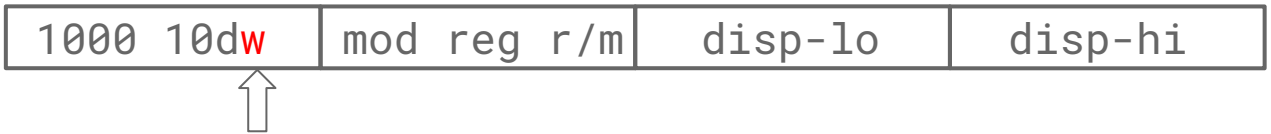
- `mov dx, [var]`
  - Look in Table 2
  - Several choices — only one will work

1000 10dw	mod reg r/m	disp-lo	disp-hi

mov

mov reg/mem to reg/mem ✔	1000 10dw	mod reg r/m	disp-lo	disp-hi		
mov immed to mem ✘	1100 011w	mod 000 r/m	disp-lo	disp-hi	data-lo	data-hi
mov immed to reg ✘	1011 w reg	data-lo	data-hi			
mov mem to accumulator ✘	1010 000w	disp-lo	disp-hi			
mov accumulator to mem ✘	1010 001w	disp-lo	disp-hi			
mov reg/mem to seg reg ✘	1000 1110	mod 0 sr r/m	disp-lo	disp-hi		
mov seg reg to reg/mem ✘	1000 1100	mod 0 sr r/m	disp-lo	disp-hi		

Example 3

- `mov dx, [var]`
  - Look in Table 2
  - Several choices — only one will work
  - 

data size is word (dx)
⇒ w = 1

Example 3

- `mov dx, [var]`
  - Look in Table 2
  - Several choices — only one will work
  - 

Example 3

- `mov dx, [var]`
  - Look in Table 2
  - Several choices — only one will work
  - 

Table 1

OPERAND(S)	MOD	REG	R/M	DISP	IMM
1. single register	11	special code	register		
2. single memory	00	special code	110	mem addr	
3. immediate to register	11	special code	register		data
4. immediate to memory	00	special code	110	mem addr	data
5. register and memory	00	register	110	mem addr	
6. register and register	11	register_1	register_2		

Example 3

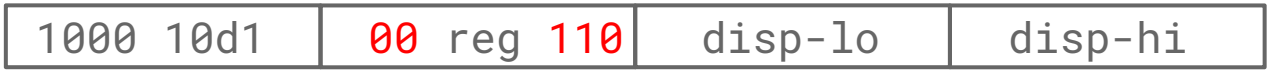
- `mov dx, [var]`
  - Look in Table 2
  - Several choices — only one will work
  - 

Table 1

OPERAND(S)	MOD	REG	R/M	DISP	IMM
1. single register	11	special code	register		
2. single memory	00	special code	110	mem addr	
3. immediate to register	11	special code	register		data
4. immediate to memory	00	special code	110	mem addr	data
5. register and memory	00	register	110	mem addr	
6. register and register	11	register_1	register_2		

Example 3

- `mov dx, [var]`
  - Look in Table 2
  - Several choices — only one will work
  - 

Table 1

OPERAND(S)	MOD	REG	R/M	DISP	IMM
1. single register	11	special code	register		
2. single memory	00	special code	110	mem addr	
3. immediate to register	11	special code	register		data
4. immediate to memory	00	special code	110	mem addr	data
5. register and memory	00	register	110	mem addr	
6. register and register	11	register_1	register_2		

Example 3

- `mov dx, [var]`
  - Look in Table 2
  - Several choices — only one will work
  - 

Table 1

Word	Byte
000 AX	000 AL
001 CX	001 CL
010 DX	010 DL
011 BX	011 BL
100 SP	100 AH
101 BP	101 CH
110 SI	110 DH
111 DI	111 BH

Example 3

- `mov dx, [var]`
  - Look in Table 2
  - Several choices — only one will work
  - 

Table 1

Word	Byte
000 AX	000 AL
001 CX	001 CL
010 DX	010 DL
011 BX	011 BL
100 SP	100 AH
101 BP	101 CH
110 SI	110 DH
111 DI	111 BH

Example 3

- `mov dx, [var]`
  - Look in Table 2
  - Several choices — only one will work
  - 

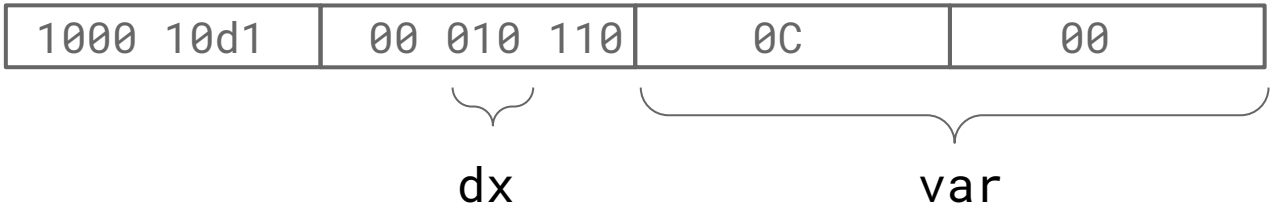
Displacement was given as 12_{10}.
Hex = 00 0C

Example 3

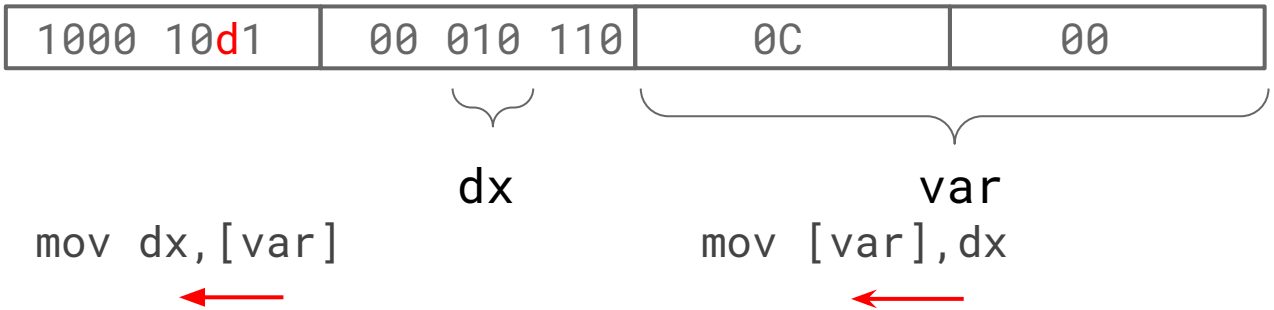
- `mov dx, [var]`
  - Look in Table 2
  - Several choices — only one will work
  - 

Displacement was given as 12_{10}.
Hex = 00 0C

Example 3

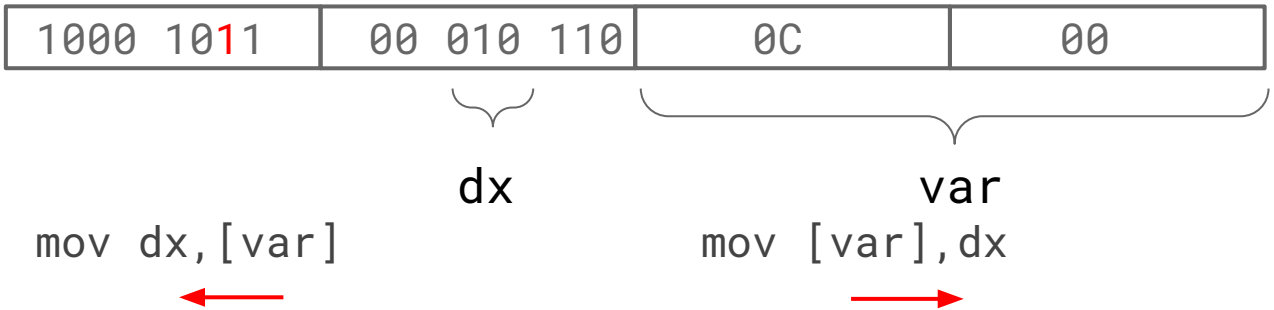
- `mov dx, [var]`
  - Look in Table 2
  - Several choices — only one will work
  - 

Example 3

- `mov dx, [var]`
  - Look in Table 2
  - Several choices — only one will work
  - 


```
D   Direction of data movement. Only needed for 2-operand instructions
    (register & memory) or (register & register). See OPERANDS 5 and 6.
    d = 0 REG field in ADDR is the source operand
    d = 1 REG field in ADDR is the destination operand
```

Example 3

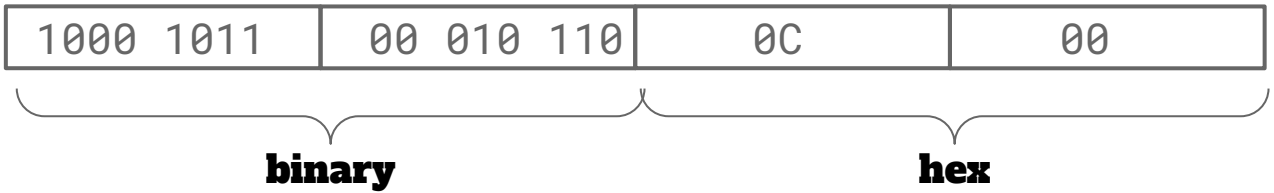
- `mov dx, [var]`
  - Look in Table 2
  - Several choices — only one will work
  - 


```
D   Direction of data movement. Only needed for 2-operand instructions
    (register & memory) or (register & register). See OPERANDS 5 and 6.
    d = 0 REG field in ADDR is the source operand
    d = 1 REG field in ADDR is the destination operand
```

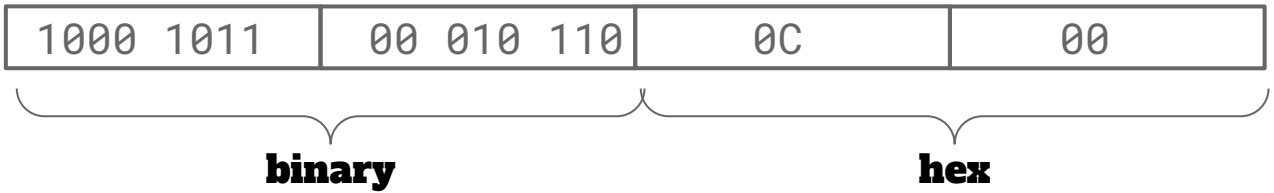
Example 3

- `mov dx, [var]`
  - Look in Table 2
  - Several choices — only one will work
  - 

Example 3

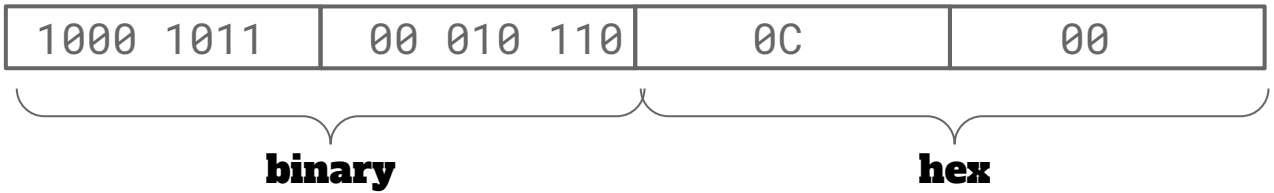
- `mov dx, [var]`
  - Look in Table 2
  - Several choices — only one will work
  - 

Example 3

- `mov dx, [var]`
  - Look in Table 2
  - Several choices — only one will work
  - 

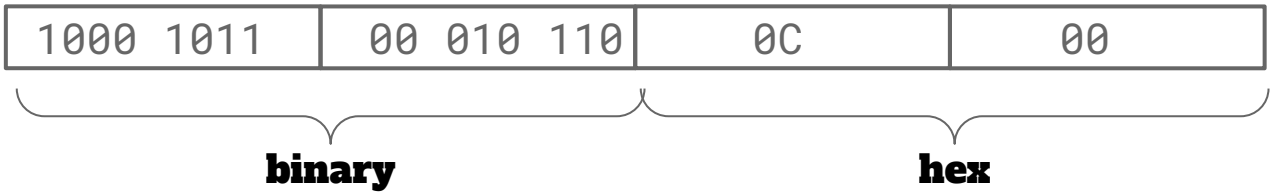
  - `1000 1011 0001 0110` — regroup

Example 3

- `mov dx, [var]`
  - Look in Table 2
  - Several choices — only one will work
  - 

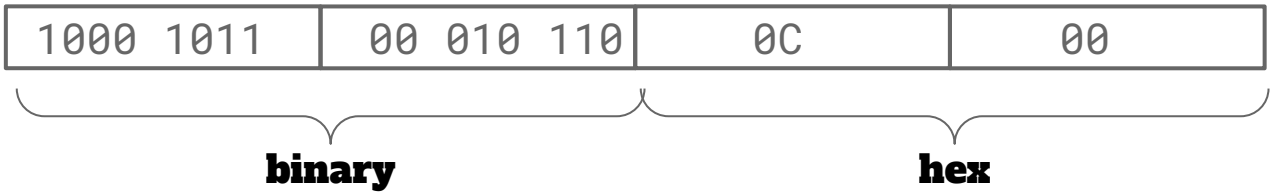
  - `1000 1011 0001 0110` — regroup
  - `8`

Example 3

- `mov dx, [var]`
  - Look in Table 2
  - Several choices — only one will work
  - 

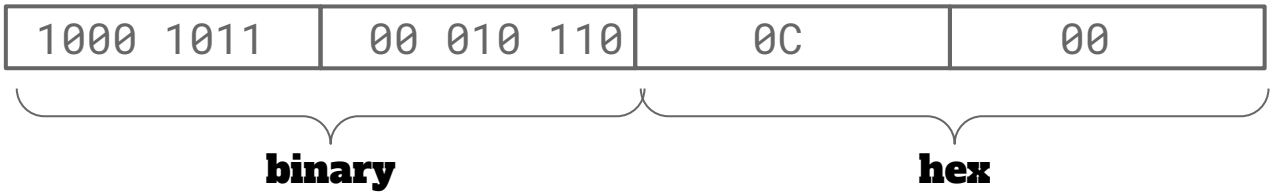
  - `1000 1011 0001 0110` — regroup
  - `8 B`

Example 3

- `mov dx, [var]`
  - Look in Table 2
  - Several choices — only one will work
  - 

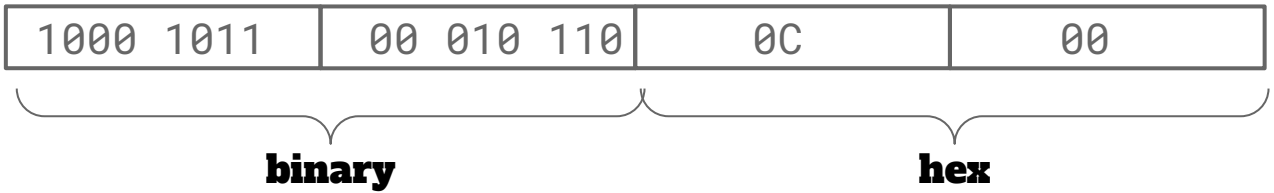
  - `1000 1011 0001 0110` — regroup
  - `8 B 1`

Example 3

- `mov dx, [var]`
  - Look in Table 2
  - Several choices — only one will work
  - 

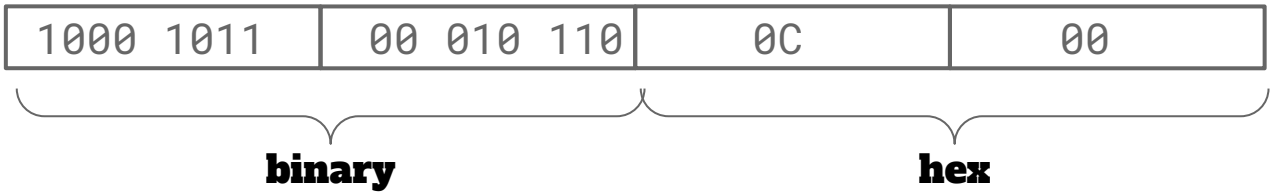
  - `1000 1011 0001 0110` — regroup
  - `8 B 1 6`

Example 3

- `mov dx, [var]`
  - Look in Table 2
  - Several choices — only one will work
  - 

  - `1000 1011 0001 0110` — regroup
  - `8B 16`

Example 3

- `mov dx, [var]`
  - Look in Table 2
  - Several choices — only one will work
  - 

  - `1000 1011 0001 0110` — regroup
  - `8B 16 0C 00`

Problem 1

- Convert to machine code
  - `dec [var]`
  - Var is a byte at offset 1000_{16}

Try it.

Problem 1

- Convert to machine code
  - `dec [var]`
  - Var is a byte at offset 1000_{16}
- From table 2
  - dec 8 bit reg / any size mem

1111 111w	mod 001 r/m	disp-lo	disp-hi

- From table 1

OPERAND(S)	MOD	REG	R/M	DISP	IMM
1. single register	11	special code	register		
2. single memory	00	special code	110	mem addr	
3. immediate to register	11	special code	register		data
4. immediate to memory	00	special code	110	mem addr	data
5. register and memory	00	register	110	mem addr	
6. register and register	11	register_1	register_2		

Problem 1

- Convert to machine code
  - `dec [var]`
  - Var is a byte at offset 1000_{16}
- From table 2
  - dec ~~8 bit reg~~ / any size mem

1111 111w	mod 001 r/m	disp-lo	disp-hi

- From table 1

OPERAND(S)	MOD	REG	R/M	DISP	IMM
1. single register	11	special code	register		
2. single memory	00	special code	110	mem addr	
3. immediate to register	11	special code	register		data
4. immediate to memory	00	special code	110	mem addr	data
5. register and memory	00	register	110	mem addr	
6. register and register	11	register_1	register_2		

Problem 1

- Convert to machine code
  - `dec [var]`
  - Var is a byte at offset 1000_{16}
- From table 2
  - dec ~~8 bit reg~~ / any size mem

0

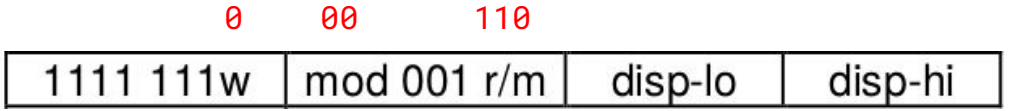
1111 111w	mod 001 r/m	disp-lo	disp-hi

- From table 1

OPERAND(S)	MOD	REG	R/M	DISP	IMM
1. single register	11	special code	register		
2. single memory	00	special code	110	mem addr	
3. immediate to register	11	special code	register		data
4. immediate to memory	00	special code	110	mem addr	data
5. register and memory	00	register	110	mem addr	
6. register and register	11	register_1	register_2		

Problem 1

- Convert to machine code
  - `dec [var]`
  - Var is a byte at offset 1000_{16}
- From table 2
  - dec ~~8 bit reg~~ / any size mem

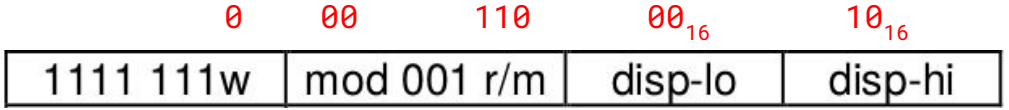


- From table 1

OPERAND(S)	MOD	REG	R/M	DISP	IMM
1. single register	11	special code	register		
2. single memory	00	special code	110	mem addr	
3. immediate to register	11	special code	register		data
4. immediate to memory	00	special code	110	mem addr	data
5. register and memory	00	register	110	mem addr	
6. register and register	11	register_1	register_2		

Problem 1

- Convert to machine code
  - `dec [var]` → `FE 0E 00 10`
  - Var is a byte at offset 1000_{16}
- From table 2
  - dec ~~8 bit reg~~ / any size mem



- From table 1

OPERAND(S)	MOD	REG	R/M	DISP	IMM
1. single register	11	special code	register		
2. single memory	00	special code	110	mem addr	
3. immediate to register	11	special code	register		data
4. immediate to memory	00	special code	110	mem addr	data
5. register and memory	00	register	110	mem addr	
6. register and register	11	register_1	register_2		

Problem 2

- Convert
  - 24 24

Try it.

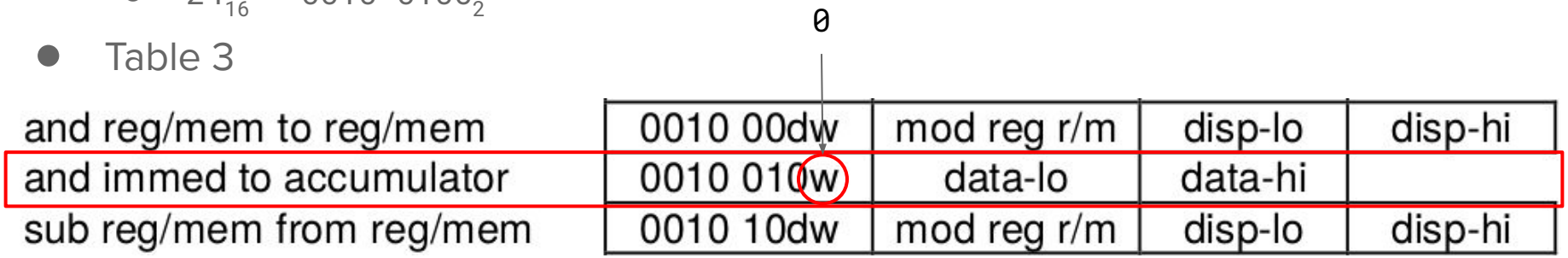
Problem 2

- Convert
  - `24 24`
  - 24_{16} = $0010\ 0100_2$
- Table 3

and reg/mem to reg/mem	0010 00dw	mod reg r/m	disp-lo	disp-hi
and immed to accumulator	0010 010w	data-lo	data-hi	
sub reg/mem from reg/mem	0010 10dw	mod reg r/m	disp-lo	disp-hi

Problem 2

- Convert
  - 24 24
  - 24_{16} = $0010\ 0100_2$
- Table 3



and reg/mem to reg/mem	0010 00dw	mod reg r/m	disp-lo	disp-hi
and immed to accumulator	0010 010w	data-lo	data-hi	
sub reg/mem from reg/mem	0010 10dw	mod reg r/m	disp-lo	disp-hi

Problem 2

- Convert
  - `24 24`
  - 24_{16} = $0010\ 0100_2$
- Table 3

`0` ⇒ **byte**

and reg/mem to reg/mem	0010 00dw	mod reg r/m	disp-lo	disp-hi
and immed to accumulator	0010 010w	data-lo	data-hi	
sub reg/mem from reg/mem	0010 10dw	mod reg r/m	disp-lo	disp-hi

Problem 2

- Convert
  - `24 24` → `and al,24h`
  - $24_{16} = 0010\ 0100_2$
- Table 3

`0` ⇒ **byte**

and reg/mem to reg/mem	0010 00dw	mod reg r/m	disp-lo	disp-hi
and immed to accumulator	0010 010w	data-lo	data-hi	
sub reg/mem from reg/mem	0010 10dw	mod reg r/m	disp-lo	disp-hi