CAO: Lecture 29
Instructions of 8086

Topics Covered

- 8086 instruction set
- Addressing modes
- Data Transfer Instructions
- Logical Instructions
- Shift and Rotate Instructions
- Arithmetic Instructions
- Transfer Instructions
- Loop Control
- String Instructions
- Repeat instructions
- Processor Control Instructions

8086 instruction set:

- **Data moving instructions.**
- **Arithmetic** - add, subtract, increment, decrement, convert byte/word and compare.
- **Logic** - AND, OR, exclusive OR, shift/rotate and test.
- **String manipulation** - load, store, move, compare and scan for byte/word.
- **Control transfer** - conditional, unconditional, call subroutine and return from subroutine.
- **Input/Output instructions**.
- Other - setting/clearing flag bits, stack operations, software interrupts, etc.

Addressing modes…

- **Implied** - the data value/data address is implicitly associated with the instruction.
- **Register** - references the data in a register or in a register pair.
- **Immediate** - the data is provided in the instruction.
- **Direct** - the instruction operand specifies the memory address where data is located.
- **Register indirect** - instruction specifies a register containing an address, where data is located. This addressing mode works with SI, DI, BX and BP registers.

Addressing modes

- **Based** - 8-bit or 16-bit instruction operand address is added to the contents of a base register (BX or BP), the resulting value is a pointer to location where data resides.
- **Indexed** - 8-bit or 16-bit instruction operand address is added to the contents of an index register (SI or DI), the resulting value is a pointer to location where data resides.
- **Based Indexed** - the contents of a base register (BX or BP) is added to the contents of an index register (SI or DI), the resulting value is a pointer to location where data resides.
- **Based Indexed with displacement** - 8-bit or 16-bit instruction operand is added to the contents of a base register (BX or BP) and index register (SI or DI), the resulting value is a pointer to location where data resides.

Data Transfer Instructions

- **MOV** :Move byte or word to register or memory
- **IN, OUT**: Input byte or word from port, output word to port
- **LEA**: Load effective address
- **LDS, LES** Load pointer using data segment, extra segment
- **PUSH, POP:** Push word onto stack, pop word off stack
- **XCHG:** Exchange byte or word
- **XLAT:** Translate byte using look-up table

Logical Instructions

- NOT : Logical NOT of byte or word (one's complement)
- AND: Logical AND of byte or word
- OR: Logical OR of byte or word
- XOR: Logical exclusive-OR of byte or word
- TEST: Test byte or word (AND without storing)

Shift and Rotate Instructions

- SHL, SHR Logical shift left, right byte or word? by 1 or CL
- SAL, SAR Arithmetic shift left, right byte or word? by 1 or CL
- ROL, ROR Rotate left, right byte or word? by 1 or CL
- RCL, RCR Rotate left, right through carry byte or word? by 1 or CL

Arithmetic Instructions

- **ADD, SUB:** Add, subtract byte or word
- **ADC, SBB :**Add, subtract byte or word and carry (borrow)
- **INC, DEC:** Increment, decrement byte or word
- **NEG:** Negate byte or word (two's complement)
- **CMP:** Compare byte or word (subtract without storing)
- **MUL, DIV:** Multiply, divide byte or word (unsigned)
- **IMUL, IDIV:** Integer multiply, divide byte or word (signed)
- **CBW, CWD:** Convert byte to word, word to double word (useful before multiply/divide)
- **AAA, AAS, AAM,AAD:** ASCII adjust for addition, subtraction, multiplication, division (ASCII codes 30-39)
- **DAA, DAS:** Decimal adjust for addition, subtraction (binary coded decimal numbers)

Transfer Instructions

JMP: Unconditional jump (*short* ?127/8, *near* ?32K, *far* between segments)

Conditional jumps:

- **JA (JNBE):** Jump if above (not below or equal)? +127, -128 range only
- **JAE (JNB):** Jump if above or equal(not below)? +127, -128 range only
- **JB (JNAE):** Jump if below (not above or equal)? +127, -128 range only
- **JBE (JNA):** Jump if below or equal (not above)? +127, -128 range only
- **JE (JZ):** Jump if equal (zero)? +127, -128 range only
- **JG (JNLE):** Jump if greater (not less or equal)? +127, -128 range only
- **JGE (JNL):** Jump if greater or equal (not less)? +127, -128 range only
- **JL (JNGE):** Jump if less (not greater nor equal)? +127, -128 range only
- **JLE (JNG):** Jump if less or equal (not greater)? +127, -128 range only
- **JC, JNC:** Jump if carry set, carry not set? +127, -128 range only
- **JO, JNO:** Jump if overflow, no overflow? +127, -128 range only
- **JS, JNS:** Jump if sign, no sign? +127, -128 range only
- **JNP (JPO):** Jump if no parity (parity odd)? +127, -128 range only
- **JP (JPE):** Jump if parity (parity even)? +127, -128 range only

Loop Control

- **LOOP:** Loop unconditional, count in CX, short jump to target address
- **LOOPE (LOOPZ):** Loop if equal (zero), count in CX, short jump to target address
- **LOOPNE (LOOPNZ):** Loop if not equal (not zero), count in CX, short jump to target address
- **JCXZ:** Jump if CX equals zero (used to skip code in loop)

Subroutine and Interrupt Instructions

- **CALL, RET:** Call, return from procedure (inside or outside current segment)
- **INT, INTO:** Software interrupt, interrupt if overflow
- **IRET:** Return from interrupt

String Instructions

- **MOVS:** Move byte or word string
- **MOVSB, MOVSW:** Move byte, word string
- **CMPS:** Compare byte or word string
- **SCAS S**: can byte or word string (comparing to A or AX)
- **LODS, STOS:** Load, store byte or word string to AL .

Repeat instructions

Repeat instructions (placed in front of other string operations):

- **REP:** Repeat
- **REPE, REPZ:** Repeat while equal, zero
- **REPNE, REPNZ:** Repeat while not equal (zero)

Processor Control Instructions

Flag manipulation:

- **STC, CLC, CMC:** Set, clear, complement carry flag
- **STD, CLD:** Set, clear direction flag
- **STI, CLI:** Set, clear interrupt enable flag
- **PUSHF, POPF:** Push flags onto stack, pop flags off stack

Coprocessor, multiprocessor interface:

- ESC Escape to external processor interface
- LOCK Lock bus during next instruction
- *Inactive states:*
- NOP No operation
- WAIT Wait for TEST pin activity
- HLT Halt processor