

- Instructions can have 0, 1 or 2 operands
 - Zero operand examples:
 - PUSHAD pushes all 8 data regs onto stack
 - POPAD restores all 8 regs from stack (popping)
 - One operand examples:
 - INC EDI adds 1 to EDI register
 - DEC EBX subtracts 1 from EBX register
 - Two operand examples:
 - ADD EAX,EBX
 - MUL EBX,511

- Load-effective-address instruction
 - LEA EAX,LOC1
 - Puts address corresponding to LOC1 into EAX
 - Address is part of the instruction
 - MOV EAX,OFFSET LOC1 has same effect
 - Address of LOC1 is known by assembler
 - LEA EBX,[EBP + 8]
 - Puts address = (contents of EBP) + 8 into EBX at runtime
 - Assembler can't know what EBP will contain

- Arithmetic instructions
 - May use all register operands
 - one operand may be in memory
 - Operands can be 8-bit or 32-bit
 - Examples:
 - `ADD EAX,EBX`
 - `ADC EAX,EBX`
 - `SUB EBX,[EAX+4]`
 - `SBB EAX,EBX`
 - `CMP [EBX + 10],AL`

■ Multiply instructions

- `IMUL EBX` implicit multiplicand is `EAX`
 - Computes $[EAX] * [EBX]$
 - Puts 64-bit product into `EDX,EAX` (high,low)
- `IMUL EBX,[EBP]`
 - Puts low 32 bits of product $dest * src$ into `dest`
 - OF flag = 1 if high half of 64-bit product is nonzero
- `IMUL` does signed multiply
- `MUL` does unsigned multiply
- Source can be immediate, register or in memory
- Destination must be a register

■ Division instructions

■ IDIV *src*

- Divides EDX,EAX pair by *src*
- 32-bit value in EAX must be sign extended to 64-bits
 - CDQ converts EAX into 64 bits in EDX,EAX
- Sets EAX=quotient and sets EDX=remainder
- Division by zero causes an exception

■ DIV does unsigned divide

■ Source can be immediate, register or in memory

- Conditional Jump instructions
 - All branches are called “jumps” with IA-32
 - Tests condition codes to decide

Mnemonic	Condition name	Condition test
JS	Sign (negative)	SF = 1
JNS	No sign (positive or zero)	SF = 0
JE/JZ	Equal/Zero	ZF = 1
JNE/JNZ	Not equal/Not zero	ZF = 0
JO	Overflow	OF = 1
JNO	No overflow	OF = 0
JC/JB	Carry/Unsigned below	CF = 1
JNC/JAE	No carry/Unsigned above or equal	CF = 0
JA	Unsigned above	$CF \vee ZF = 0$
JBE	Unsigned below or equal	$CF \vee ZF = 1$
JGE	Signed greater than or equal	$SF \oplus OF = 0$
JL	Signed less than	$SF \oplus OF = 1$
JG	Signed greater than	$ZF \vee (SF \oplus OF) = 0$
JLE	Signed less than or equal	$ZF \vee (SF \oplus OF) = 1$


\vee denotes OR

\oplus denotes XOR

E.g.: JG EXIT

- Conditional Jump instructions
 - Assembler generates signed offset relative to next location
 - One-byte offset if in the range -128 to +127
 - Otherwise a 4-byte offset is generated
 - $IP + \text{Offset} = \text{target address}$
- Loop Instruction

START:	MOV	ECX,NUM_PASSES		START:	MOV	ECX,NUM_PASSES
.				.		
.				.		
.				.		
DEC	ECX			LOOP	START	
JG	START					



LOOP instruction implicitly decrements ECX and branches if > 0

- JMP is the unconditional jump
 - Uses 1-byte or 4-byte signed offset
 - May also use other addressing modes for target address
 - Example: JMP [JUMPTBLE + ESI * 4]
 - Uses ESI as Index into table of jump addresses
 - Implements high-level language Case statement