The following pages contain references for use during the exam: tables containing the x86 instruction set (covered so far) and condition codes. You do not need to submit these pages when you finish your exam.

Remember that:

- Most instructions can have at most one memory operand.
- Brackets [ ] around a register name, immediate, or combination of the two indicates an effective address.
    - Example: MOV AX, [0x10] à contents of address 0x10 moved to AX
- Parentheses around an address mean “the contents of memory at this address”.
    - Example: (0x10) à the contents of memory at address 0x10

Category	**Instruction**	**Example**	**Meaning**
Data transfer	Move	`MOV AX, BX`	`AX = BX`
	Move & sign-extend	`MOVSX EAX, DL`	`EAX = DL, sign-extended to 32 bits`
	Move and zero-extend	`MOVZX EAX, DL`	`EAX = DL, zero-extended to 32 bits`
	Exchange	`XCHG AX, BX`	`Swap contents of AX, BX`
	Load effective address	`LEA AX,[BX+SI+0x10]`	`AX = BX + SI + 0x10`
Arithmetic	Add	`ADD AX, BX`	`AX = AX + BX`
	Add with carry	`ADC AX, BX`	`AX = AX + BX + CF`
	Increment	`INC [EDI]`	`(EDI) = (EDI) + 1`
	Subtract	`SUB AX, [0x10]`	`AX = AX – (0x10)`
	Subtract with borrow	`SBB AX, [0x10]`	`AX = AX – (0x10) – CF`
	Decrement	`DEC CX`	`CX = CX – 1`
	Negate (2’s complement)	`NEG CX`	`CX = -CX`

	Multiply
	Unsigned: `MUL`
	(all operands are non-negative)
	Signed: `IMUL`
	(all operands are signed integers in 2’s complement form)
		`IMUL BH`
		
		`IMUL CX`
		
		`MUL DWORD PTR [0x10]`
			`AX = BH * AL`
			
			`(DX,AX) = CX * AX`
			
			`(EDX,EAX) = (0x10) * EAX`

	Divide
	Unsigned: `DIV`
	(all operands are non-negative)
	Signed: `IDIV`
	(all operands are signed integers in 2’s complement form)
		`DIV BH`
		
		`IDIV CX`
		
		`DIV EBX`
			`AL = AX / BH (quotient)`
			`AH = AX % BH (remainder)`
			
			`AX = EAX / CX (quotient)`
			`DX = EAX % CX (remainder)`
			
			`EAX = (EDX,EAX) / EBX (Q)`
			`EDX = (EDX,EAX) % EBX (R)`