## 8086 INSTRUCTION REFERENCE

1		land land and	75	
	lest, src	dest = dest + src	JMP target	unconditional jump
Flags: ZF = 1 iff result = 0 SF = 1 iff dest < 0 CF = 1 iff result generates carry out of most signif. bit		<pre>IP = IP + offset_to_target Addressing modes: target uses relative addressing</pre>		
OF = 1 iff result creates signed overflow		Flags: no effect		
			<b></b>	
CMP d	lest, src	calc dest - src	MOV dest, src	dest = src
	es <b>not</b> modify dest, modifies FL		Flags: no effect	
	ZF = 1 iff result = 0 SF = 1 iff re			
	CF = 1 iff result requires borrow DF = 1 iff result creates signed of		MUL src	unsigned integer multiplication
	Dr = 1 III lesuit creates signed t	overnow	if 8-bit source: AX := AL * if 16-bit source: DX:AX :=	
DIV s	erc uns	signed integer divide		ay be reg/mem8 or reg/mem16
if 8-bit s		AL = AX / src (division)	NB: src may not be imme	
	$AH = AX \mod src (rem$	ainder)	Flags: $CF = 0$ iff $AH = 0$	(for 8-bit src)
if 16-bit		DX:AX / src (division)		(for 16-bit src)
DX = AX mod src (remainder)			OF = CF	
Addressing modes: src may be reg/mem8 or reg/mem16 <b>NB</b> : src may <b>not</b> be immediate!			ZF and SF are undefined following MUL	
	he flags are undefined following	g DIV	NEG dest	dest = 0 - dest
•	· ·	•	Addressing modes: dest r	nay be 8 or 16 reg/mem
DEC d	lest	dest = dest - 1	Flags: CF = 0 iff result = 0	
INC dest = dest + 1			OF = 1 iff value could not be negated correctly	
Addressing modes: dest may be reg/mem8 or reg/mem16  Flags: ZF = 1 iff result = 0 SF = 1 iff result < 0			OUT [DX], AL	port[DX] = AL
	CF = 1 iii result = 0 SF = 1 iii res CF = 1 iff result generates carry		Addressing modes: only those shown (indirect DX)	
	or = 1 in result generates earry	most signif. bit	Flags: no effect	nose shown (maneer bx)
_	OF 4 :44		go:	
(	OF = 1 iff result creates signed of	overnow		
C	OF = 1 lift result creates signed (	overnow	POP dest	pop from stack
	-		dest := mem[SP]	pop from stack
Jcc t	arget	conditional jump	dest := mem[SP] SP := SP + 2	
Jcc t IP:= IP -	carget + offset_to_target (if condition <b>t</b>	conditional jump rue)	dest := mem[SP] SP := SP + 2 Addressing modes: dest r	pop from stack may be 16 bit reg/mem only
Jcc to the second secon	carget + offset_to_target (if condition t sing modes: target uses short re	conditional jump rue)	dest := mem[SP] SP := SP + 2	
Jcc t IP:= IP - Address Flags: r	carget + offset_to_target (if condition <b>t</b>	conditional jump rue) elative addressing	dest := mem[SP] SP := SP + 2 Addressing modes: dest r	
Jcc t IP:= IP - Address Flags: r The pos	carget  + offset_to_target (if condition to sing modes: target uses short reduced to effect saible forms are summarized be	conditional jump rue) elative addressing elow and to the right:	<pre>dest := mem[SP] SP := SP + 2 Addressing modes: dest r Flags: no effect  PUSH src SP := SP - 2</pre>	nay be 16 bit reg/mem <b>only</b>
Jcc to the second secon	carget  + offset_to_target (if condition the sing modes: target uses short received to effect saidle forms are summarized be summarized.	conditional jump rue) elative addressing elow and to the right: = 0 and ZF = 0 (JNBE)	<pre>dest := mem[SP] SP := SP + 2 Addressing modes: dest r Flags: no effect  PUSH src SP := SP - 2 mem[SP] := src</pre>	nay be 16 bit reg/mem <b>only</b> push onto stack
Jcc to the second secon	carget  + offset_to_target (if condition to sing modes: target uses short represented by the same of t	conditional jump rue) elative addressing elow and to the right: = 0 and ZF = 0 (JNBE) or Equal CF = 0 (JNB)	<pre>dest := mem[SP] SP := SP + 2 Addressing modes: dest r Flags: no effect  PUSH src SP := SP - 2 mem[SP] := src Addressing modes: src m</pre>	nay be 16 bit reg/mem <b>only</b>
Jcc to the second secon	carget  + offset_to_target (if condition to sing modes: target uses short remote effect saible forms are summarized be sump Above CF:  Jump Above  Jump Above	conditional jump rue) elative addressing elow and to the right: = 0 and ZF = 0 (JNBE) or Equal CF = 0 (JNB) b Below CF = 1 (JNAE)	<pre>dest := mem[SP] SP := SP + 2 Addressing modes: dest r Flags: no effect  PUSH src SP := SP - 2 mem[SP] := src</pre>	nay be 16 bit reg/mem <b>only</b> push onto stack
Jcc to the second secon	carget  + offset_to_target (if condition to sing modes: target uses short represented by the same of t	conditional jump rue) elative addressing elow and to the right: = 0 and ZF = 0 (JNBE) or Equal CF = 0 (JNB) b Below CF = 1 (JNAE) CF = 1 or ZF = 1 (JNA)	dest := mem[SP] SP := SP + 2 Addressing modes: dest r Flags: no effect  PUSH src SP := SP - 2 mem[SP] := src Addressing modes: src m Flags: no effect	nay be 16 bit reg/mem <b>only</b> push onto stack
Jcc to the second secon	carget  + offset_to_target (if condition to sing modes: target uses short remote effect saible forms are summarized be sump Above CF:  Jump Above  Jump Above	conditional jump rue) elative addressing elow and to the right: = 0 and ZF = 0 (JNBE) or Equal CF = 0 (JNB) b Below CF = 1 (JNAE) CF = 1 or ZF = 1 (JNA) Jump Carry CF = 1	<pre>dest := mem[SP] SP := SP + 2 Addressing modes: dest r Flags: no effect  PUSH src SP := SP - 2 mem[SP] := src Addressing modes: src m Flags: no effect  SHL dest,CL dest</pre>	push onto stack  ay be 16 bit reg/mem only  = dest shifted_left_by CL
Jcc to IP:= IP:- Address Flags: r The post JA JAE JB JBE JC JE	carget  + offset_to_target (if condition to sing modes: target uses short remote effect saible forms are summarized become Jump Above CF:  Jump Above  Jump Jump Below or Equal C	conditional jump rue) elative addressing elow and to the right: = 0 and ZF = 0 (JNBE) or Equal CF = 0 (JNB) b Below CF = 1 (JNAE) CF = 1 or ZF = 1 (JNA) Jump Carry CF = 1 Jump Equal ZF = 1	dest := mem[SP] SP := SP + 2 Addressing modes: dest r Flags: no effect  PUSH src SP := SP - 2 mem[SP] := src Addressing modes: src m Flags: no effect  SHL dest, CL dest Addressing modes: dest r	push onto stack  ay be 16 bit reg/mem only  = dest shifted_left_by CL = dest shifted_right_by CL  ay be reg/mem8 or reg/mem16
Jcc to the second secon	earget  + offset_to_target (if condition to sing modes: target uses short remote effect assible forms are summarized become a	conditional jump rue) elative addressing elow and to the right: = 0 and ZF = 0 (JNBE) or Equal CF = 0 (JNBE) b Below CF = 1 (JNAE) CF = 1 or ZF = 1 (JNA) Jump Carry CF = 1 Jump Equal ZF = 1 OF and ZF = 0 (JNLE)	dest := mem[SP] SP := SP + 2 Addressing modes: dest r Flags: no effect  PUSH src SP := SP - 2 mem[SP] := src Addressing modes: src m Flags: no effect  SHL dest, CL dest SHR dest, CL dest Addressing modes: dest r Flags: CF = value of bit sh	push onto stack  ay be 16 bit reg/mem only  = dest shifted_left_by CL = dest shifted_right_by CL may be reg/mem8 or reg/mem16 nifted out of reg
Jcc to the second secon	carget  + offset_to_target (if condition to sing modes: target uses short response of effect saible forms are summarized because of Jump Above CF sump Above  Jump Below or Equal County Jump Greater Than SF = 0	conditional jump rue) elative addressing elow and to the right: = 0 and ZF = 0 (JNBE) or Equal CF = 0 (JNB) DEBELOW CF = 1 (JNAE) DEFELOW CF = 1 (JNAE) Jump Carry CF = 1 Jump Equal ZF = 1 OF and ZF = 0 (JNLE) r Equal SF = OF (JNL)	dest := mem[SP] SP := SP + 2 Addressing modes: dest r Flags: no effect  PUSH src SP := SP - 2 mem[SP] := src Addressing modes: src m Flags: no effect  SHL dest, CL dest SHR dest, CL dest Addressing modes: dest r Flags: CF = value of bit sh OF = 1 iff result and	push onto stack  ay be 16 bit reg/mem only  = dest shifted_left_by CL = dest shifted_right_by CL may be reg/mem8 or reg/mem16 ifted out of reg d original value have different signs
Jcc to the second secon	carget  + offset_to_target (if condition to sing modes: target uses short reported by the sible forms are summarized because of the sible forms are summariz	conditional jump rue) elative addressing elow and to the right:  = 0 and ZF = 0 (JNBE) or Equal CF = 0 (JNB) b Below CF = 1 (JNAE) CF = 1 or ZF = 1 (JNA) Jump Carry CF = 1 Jump Equal ZF = 1 OF and ZF = 0 (JNLE) r Equal SF = OF (JNL) Than SF != OF (JNGE)	dest := mem[SP] SP := SP + 2 Addressing modes: dest r Flags: no effect  PUSH src SP := SP - 2 mem[SP] := src Addressing modes: src m Flags: no effect  SHL dest, CL dest SHR dest, CL dest Addressing modes: dest r Flags: CF = value of bit sh	push onto stack  ay be 16 bit reg/mem only  = dest shifted_left_by CL = dest shifted_right_by CL may be reg/mem8 or reg/mem16 ifted out of reg d original value have different signs
Jcc to the second secon	earget  + offset_to_target (if condition to sing modes: target uses short response of each of the sible forms are summarized be sump Above CF:  Jump Above	conditional jump rue) elative addressing elow and to the right: = 0 and ZF = 0 (JNBE) or Equal CF = 0 (JNB) DEBELOW CF = 1 (JNAE) DEFELOW CF = 1 (JNAE) Jump Carry CF = 1 Jump Equal ZF = 1 OF and ZF = 0 (JNLE) r Equal SF = OF (JNL) Than SF!= OF (JNGE) != OF or ZF = 1 (JNG)	dest := mem[SP] SP := SP + 2 Addressing modes: dest r Flags: no effect  PUSH src SP := SP - 2 mem[SP] := src Addressing modes: src m Flags: no effect  SHL dest, CL dest SHR dest, CL dest Addressing modes: dest r Flags: CF = value of bit sh OF = 1 iff result and ZF and SF reflect th	push onto stack  ay be 16 bit reg/mem only  = dest shifted_left_by CL = dest shifted_right_by CL may be reg/mem8 or reg/mem16 ifted out of reg d original value have different signs he result
Jcc to the second secon	carget  + offset_to_target (if condition to sing modes: target uses short reported by the sible forms are summarized because of the sible forms are summariz	conditional jump rue) elative addressing elow and to the right: = 0 and ZF = 0 (JNBE) or Equal CF = 0 (JNB) DEF = 1 or ZF = 1 (JNA) Jump Carry CF = 1 Jump Equal ZF = 1 OF and ZF = 0 (JNLE) r Equal SF = OF (JNL) Than SF!= OF (JNGE) != OF or ZF = 1 (JNG) Jump No Carry CF = 0	dest := mem[SP] SP := SP + 2 Addressing modes: dest r Flags: no effect  PUSH src SP := SP - 2 mem[SP] := src Addressing modes: src m Flags: no effect  SHL dest, CL dest SHR dest, CL dest Addressing modes: dest r Flags: CF = value of bit sh OF = 1 iff result and ZF and SF reflect th	push onto stack  ay be 16 bit reg/mem only  = dest shifted_left_by CL = dest shifted_right_by CL may be reg/mem8 or reg/mem16 nifted out of reg d original value have different signs he result  dest := dest - src
Jcc to the second secon	carget  + offset_to_target (if condition to sing modes: target uses short reported by the sible forms are summarized because of the sible forms are summariz	conditional jump rue) elative addressing elow and to the right:  = 0 and ZF = 0 (JNBE) or Equal CF = 0 (JNB) b Below CF = 1 (JNAE) CF = 1 or ZF = 1 (JNA) Jump Carry CF = 1 Jump Equal ZF = 1 OF and ZF = 0 (JNLE) r Equal SF = OF (JNL) Than SF!= OF (JNGE) != OF or ZF = 1 (JNG) Jump No Carry CF = 0 ump Not Equal ZF = 0	dest := mem[SP] SP := SP + 2 Addressing modes: dest r Flags: no effect  PUSH src SP := SP - 2 mem[SP] := src Addressing modes: src m Flags: no effect  SHL dest, CL dest SHR dest, CL dest Addressing modes: dest r Flags: CF = value of bit sh OF = 1 iff result and ZF and SF reflect th  SUB dest, src Flags: ZF = 1 iff result = 0	push onto stack  ay be 16 bit reg/mem only  = dest shifted_left_by CL = dest shifted_right_by CL may be reg/mem8 or reg/mem16 nifted out of reg d original value have different signs he result  dest := dest - src  O SF = 1 iff result < 0
Jcc to IP:= IP:= Address Flags: rather post JA JAE JB JBE JC JE JC JG JGE JL JLE JNC JNC JNE JNO	carget  + offset_to_target (if condition to sing modes: target uses short reported by the sible forms are summarized because of the sible forms are summariz	conditional jump rue) elative addressing elow and to the right: = 0 and ZF = 0 (JNBE) or Equal CF = 0 (JNB) DEBELOW CF = 1 (JNAE) DEFELOW CF = 1 (JNAE) UMP Carry CF = 1 Jump Equal ZF = 1 OF and ZF = 0 (JNLE) TEQUAL SF = OF (JNLE) TEQUAL SF = OF (JNGE) != OF or ZF = 1 (JNGE) Jump No Carry CF = 0 ump Not Equal ZF = 0 up No Overflow OF = 0	dest := mem[SP] SP := SP + 2 Addressing modes: dest r Flags: no effect  PUSH src SP := SP - 2 mem[SP] := src Addressing modes: src m Flags: no effect  SHL dest, CL dest SHR dest, CL dest Addressing modes: dest r Flags: CF = value of bit sh OF = 1 iff result and ZF and SF reflect th  SUB dest, src Flags: ZF = 1 iff result = 0 CF = 1 iff result ge	push onto stack  ay be 16 bit reg/mem only  = dest shifted_left_by CL = dest shifted_right_by CL may be reg/mem8 or reg/mem16 nifted out of reg d original value have different signs he result  dest := dest - src
Jcc to the second secon	carget  + offset_to_target (if condition to sing modes: target uses short reported by the sible forms are summarized because of the sible forms are summarized by the sible forms are summarized by th	conditional jump rue) elative addressing elow and to the right: = 0 and ZF = 0 (JNBE) or Equal CF = 0 (JNB) DESIGN ELOW ELOW ELOW ELOW ELOW ELOW ELOW ELOW	dest := mem[SP] SP := SP + 2 Addressing modes: dest r Flags: no effect  PUSH src SP := SP - 2 mem[SP] := src Addressing modes: src m Flags: no effect  SHL dest, CL dest SHR dest, CL dest Addressing modes: dest r Flags: CF = value of bit sh OF = 1 iff result and ZF and SF reflect th  SUB dest, src Flags: ZF = 1 iff result = 0 CF = 1 iff result ge	push onto stack  ay be 16 bit reg/mem only  = dest shifted_left_by CL = dest shifted_right_by CL may be reg/mem8 or reg/mem16 nifted out of reg d original value have different signs he result  dest := dest - src  O SF = 1 iff result < 0 nerates borrow into most-signif bit
Jcc to IP:= IP:= Address Flags: rather post JA JAE JB JBE JC JE JC JG JGE JL JLE JNC JNC JNE JNO	earget  + offset_to_target (if condition to sing modes: target uses short reported by the sible forms are summarized because of the sible forms are summarized by the sible forms are summarized by th	conditional jump rue) elative addressing elow and to the right: = 0 and ZF = 0 (JNBE) or Equal CF = 0 (JNB) DEBELOW CF = 1 (JNAE) DEFELOW CF = 1 (JNAE) UMP Carry CF = 1 Jump Equal ZF = 1 OF and ZF = 0 (JNLE) TEQUAL SF = OF (JNLE) TEQUAL SF = OF (JNGE) != OF or ZF = 1 (JNGE) Jump No Carry CF = 0 ump Not Equal ZF = 0 up No Overflow OF = 0	dest := mem[SP] SP := SP + 2 Addressing modes: dest r Flags: no effect  PUSH src SP := SP - 2 mem[SP] := src Addressing modes: src m Flags: no effect  SHL dest, CL dest SHR dest, CL dest Addressing modes: dest r Flags: CF = value of bit sh OF = 1 iff result and ZF and SF reflect th  SUB dest, src Flags: ZF = 1 iff result = 0 CF = 1 iff result ge	push onto stack  ay be 16 bit reg/mem only  = dest shifted_left_by CL = dest shifted_right_by CL may be reg/mem8 or reg/mem16 nifted out of reg d original value have different signs he result  dest := dest - src  O SF = 1 iff result < 0 nerates borrow into most-signif bit

Jump Sign SF = 1

Jump Zero ZF = 1

JS

JΖ