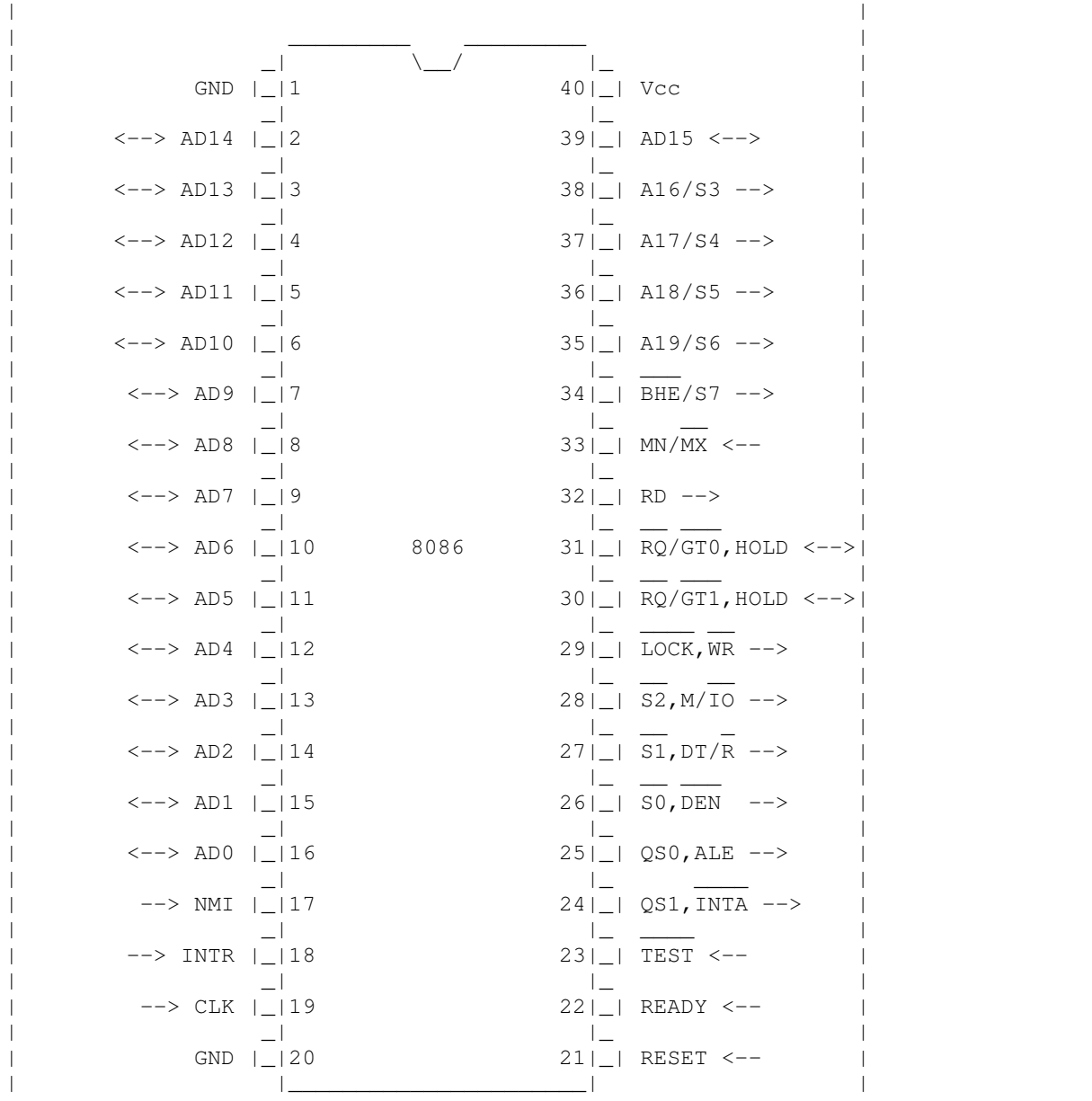
Intel

```
 88888      000      88888      666
8     8    0   0    8     8    6
8     8   0   0 0   8     8   6
 88888    0  0  0    88888    666666
8     8   0 0   0   8     8   6     6
8     8    0   0    8     8   6     6
 88888      000      88888     66666
```

8086 MICROPROCESSOR Instruction Set Summary



Written by Jonathan Bowen
Programming Research Group
Oxford University Computing Laboratory
8-11 Keble Road
Oxford OX1 3QD
England

Tel +44-865-273840

Created October 1982
Updated April 1985
Issue 1.3

Mnemonic	ODITSZAPC	Description
AAA	?---??*?*	ASCII Adjust for Add in AX
AAD	?---**?*?	ASCII Adjust for Divide in AX
AAM	?---**?*?	ASCII Adjust for Multiply in AX
AAS	?---??*?*	ASCII Adjust for Subtract in AX
ADC d,s	*---*****	Add with Carry
ADD d,s	*---*****	Add
AND d,s	*---**?**	Logical AND
CALL a	---------	Call
CBW	---------	Convert Byte to Word in AX
CLC	--------0	Clear Carry
CLD	-0-------	Clear Direction
CLI	--0------	Clear Interrupt
CMC	--------*	Complement Carry
CMP d,s	*---*****	Compare
CMPS	*---*****	Compare memory at SI and DI
CWD	---------	Convert Word to Double in AX,DX
DAA	?---*****	Decimal Adjust for Add in AX
DAS	?---*****	Decimal Adjust for Subtract in AX
DEC d	*---****-	Decrement
DIV s	?---?????	Divide (unsigned) in AX(,DX)
ESC s	---------	Escape (to external device)
HLT	---------	Halt
IDIV s	?---?????	Divide (signed) in AX(,DX)
IMUL s	*---????*	Multiply (signed) in AX(,DX)
IN d,p	---------	Input
INC d	*---****-	Increment
INT	--00-----	Interrupt
INTO	--**-----	Interrupt on Overflow
IRET	*********	Interrupt Return
JB/JNAE a	---------	Jump on Below/Not Above or Equal
JBE/JNA a	---------	Jump on Below or Equal/Not Above
JCXZ a	---------	Jump on CX Zero
JE/JZ a	---------	Jump on Equal/Zero
JL/JNGE a	---------	Jump on Less/Not Greater or Equal
JLE/JNG a	---------	Jump on Less or Equal/Not Greater
JMP a	---------	Unconditional Jump
JNB/JAE a	---------	Jump on Not Below/Above or Equal
JNBE/JA a	---------	Jump on Not Below or Equal/Above
JNE/JNZ a	---------	Jump on Not Equal/Not Zero

JNL/JGE a	---------	Jump on Not Less/Greater or Equal
JNLE/JG a	---------	Jump on Not Less or Equal/Greater
JNO a	---------	Jump on Not Overflow
JNP/JPO a	---------	Jump on Not Parity/Parity Odd
JNS a	---------	Jump on Not Sign
JO a	---------	Jump on Overflow
JP/JPE a	---------	Jump on Parity/Parity Even
JS a	---------	Jump on Sign
LAHF	---------	Load AH with 8080 Flags
LDS r,s	---------	Load pointer to DS
LEA r,s	---------	Load EA to register
LES r,s	---------	Load pointer to ES
LOCK	---------	Bus Lock prefix
LODS	---------	Load memory at SI into AX
LOOP a	---------	Loop CX times
LOOPNZ/LOOPNE a	---------	Loop while Not Zero/Not Equal
LOOPZ/LOOPE a	---------	Loop while Zero/Equal
MOV d,s	---------	Move
MOVS	---------	Move memory at SI to DI
MUL s	*---????*	Multiply (unsigned) in AX(,DX)
NEG d	*---*****	Negate
NOP	---------	No Operation (= XCHG AX,AX)
NOT d	---------	Logical NOT
OR d,s	*---**?**	Logical inclusive OR
OUT p,s	---------	Output
POP d	---------	Pop
POPF	*********	Pop Flags
PUSH s	---------	Push
PUSHF	---------	Push Flags
RCL d,c	*-------*	Rotate through Carry Left
RCR d,c	*-------*	Rotate through Carry Right
REP/REPNE/REPNZ	---------	Repeat/Repeat Not Equal/Not Zero
REPE/REPZ	---------	Repeat Equal/Zero
RET (s)	---------	Return from call
ROL d,c	--------	Rotate Left
ROR d,c	*-------*	Rotate Right
SAHF	----*****	Store AH into 8080 Flags

Mnemonic	ODITSZAPC	Description
SAR d,c	*---**?**	Shift Arithmetic Right
SBB d,s	*---*****	Subtract with Borrow
SCAS	*---*****	Scan memory at DI compared to AX
SEG r	---------	Segment register
SHL/SAL d,c	*---**?**	Shift logical/Arithmetic Left
SHR d,c	*---**?**	Shift logical Right
STC	--------1	Set Carry
STD	-0-------	Set Direction
STI	--0------	Set Interrupt
STOS	---------	Store AX into memory at DI
SUB d,s	*---*****	Subtract
TEST d,s	*---**?**	AND function to flags
WAIT	---------	Wait
XCHG r(,d)	---------	Exchange
XLAT	---------	Translate byte to AL
XOR d,s	*---**?**	Logical Exclusive OR
	-*01?	Unaff/affected/reset/set/unknown

OF	O	Overflow Flag (Bit 11)
DF	D	Direction Flag (Bit 10)
IF	I	Interrupt enable Flag (Bit 9)
TF	T	Trap Flag (Bit 8)
SF	S	Sign Flag (Bit 7)
ZF	Z	Zero Flag (Bit 6)
AF	A	Auxilary carry Flag (Bit 4)
PF	P	Parity Flag (Bit 2)
CF	C	Carry Flag (Bit 0)

ALIGN	Align to word boundary
ASSUME sr:sy(,...)	Assume segment register name(s)
ASSUME NOTHING	Remove all former assumptions
DB e(,...)	Define Byte(s)
DBS e	Define Byte Storage
DD e(,...)	Define Double Word(s)
DDS e	Define Double Word Storage
DW e(,...)	Define Word(s)
DWS e	Define Word Storage
EXT (sr:)sy(t)	External(s)(t=ABS/BYTE/DWORD/FAR/NEAR/WORD)
LABEL t	Label (t=BYTE/DWORD/FAR/NEAR/WORD)
PROC t	Procedure (t=FAR/NEAR, default NEAR)

ABS	Absolute value of operand
BYTE	Byte type operation
DWORD	Double Word operation
FAR	IP and CS registers altered
HIGH	High-order 8 bits of 16-bit value
LENGTH	Number of basic units
LOW	Low-order 8 bit of 16-bit value
NEAR	Only IP register need be altered
OFFSET	Offset portion of an address
PTR	Create a variable or label
SEG	Segment of address
SHORT	One byte for a JMP operation
SIZE	Number of bytes defined by statement
THIS	Create a variable/label of specified type
TYPE	Number of bytes in the unit defined
WORD	Word operation

AX BX CX DX	Accumulator/Base/Count/Data registers
AL BL CL DL	Low byte of general registers
AH BH CH DH	High byte of general registers
SP BP	Stack/Base Pointer registers
SI DI	Source/Destination Index registers
CS DS SS ES	Code/Data/Stack/Extra Segment registers
IP	Instruction Pointer register

a	Address
c	Count
d	Destination
e	Expression or string
p	I/O port
r	Register
s	Source
sr	Segment register (CS,DS,SS,ES)
sy	Symbol
t	Type of symbol

http://www.clipx.net/ng/iapx86/