Assemblador i386

Introducció als microprocessadors x86

Antecedents i característiques generals

Començarem aquest capítol explicant breument els antecedents d'aquesta CPU i la evolució que ha tingut. Intel va començar als principis de la dècada dels 1970 a desenvolupar el seu primer microprocessador, el **4040** de 4 bits que bàsicament estava dissenyat per petites tasques de control, l'èxit obtingut amb aquest va fer apuntar més amunt amb els objectius i va desenvolupar el microprocessador **8080** de 8 bits i amb una arquitectura i suport de perifèrics important en aquells moments, la gran demanda va fer que diversos fabricants es convertissin en segones fonts i a més van sortir al mercat varies versions basades en aquesta arquitectura, incloent el **Z80** que va mantenir la compatibilitat de software tot i que era proveït de més recursos i un joc de perifèrics propi, aquesta darrera CPU amb algunes variants encara forma part del nucli de molts microcontroladors de fabricants japonesos. Amb el Intel 8080 i el Zilog Z80 van aparèixer els primers PC amb el sistema operatiu **CP/M**, precursor del **DOS**. L'empresa ZILOG va aparèixer d'uns enginyers que es van separar de Intel, la CPU Z80 no es més que la integració del hardware de 2 CPU del 8080 i tornant a programar convenientment els recursos varen fer una CPU que mantenint la compatibilitat de software era molt més potent.

Ja consolidada en el mercat i apuntant encara més a munt, Intel va començar a desenvolupar una arquitectura per una CPU de 16 bits i va treure dues versions de CPU el **8088** i el **8086** que amb una arquitectura interna de 16 bits, poden treballar amb operant de 8 i de 16 bits, amb una capacitat d'adreçar de 20 bits, i amb el mateix joc d'instruccions. El microprocessador 8088 estava dissenyat per treballar en entorns de bus de 8 bits (duplica els accessos al bus) (comptabilitzant amb els bussos de 8 bits del 8080) i el 8086 per entorns de bus de 16 bits.

La filosofia de disseny de la família del 8086 es basa en la compatibilitat i amb la creació de sistemes informàtics integrats, i apareixen amb aquesta família una quantitat important de circuits perifèrics i coprocessadors que permeten integrar un PC de una forma relativament simple. Molts d'aquests perifèrics estan dissenyats per permetre la compatibilitat amb 8 bits, i la majoria dels seus registres tenen aquesta mida.

Aquestes CPU integrades disposen de 92 instruccions i permeten fins 7 modes d'adreçar, poden adreçar memòria de codi i dades fins a 1Mbyte i tenen instruccions especifiques d'entrada i sortida amb una capacitat d'adreçar de 64K ports, de forma que utilitza senyals de control diferents per adreçar memòria i per adreçar entrada i sortida.
Executen les instruccions entre 2 cicles teòrics de rellotge i uns 9 reals per moviments de dades entre registres interns i 206 cicles de rellotge, les més

lentes, que corresponen a la divisió entera amb signe del acumulador amb una paraula que prové de la memòria. Les freqüències típiques de rellotge de les primeres CPU son de 4.77 MHz. per el 8088 que pot realitzar entre 20000 a 500000 instruccions per segon. Per les versions de 8086 el rellotge va de 8 a 10 MHz. Aquestes CPU per el tipus d'instruccions que executen son del tipus anomenat **CISC** (Complex Instruction Set Computer), el contrari del CISC es el RISC (Reduced Instruction Set Computer).

A partir dels nuclis de 8088 i de 8086 es fan alguns petits canvis i apareixen els nous **80188** i **80186** que tindran una vida molt curta i seran usats bàsicament per sistemes de control industrial ja que son força més ràpids que els seus antecessors. NEC treu al mercat els processadors NEC-V30 i NEC-V20 que son compatibles HARDWARE i SOFTWARE amb l'avantatge de tenir algunes operacions aritmètiques tres vegades més ràpides i de tenir una cua **FIFO** de cerca prèvia.

Un salt considerable es la sortida al mercat del **80286** que es caracteritza per tenir 2 modes de funcionament completament diferenciats, el **mode real** que es el mode per defecte quan es connectat a la corrent i el **mode protegit** en el que s'ha dotat de capacitat de manegar processos multitasca i de emmagatzematge en memòria virtual. El procés multitasca consisteix en realitzar varis processos de manera aparentment simultània, amb l'ajut del sistema operatiu per commutar d'una tasca a una altra automàticament fent més òptima la utilització de la CPU, ja que mentre un procés esta esperant que un perifèric acabi una operació, pot passar el control per atendre un altra procés. La memòria virtual permet usar més memòria de la que realment es disposa, guardant part d'ella en el disc, d'aquesta manera els programes creuen que hi ha més memòria, i quan accedeixen a la memòria i no existeix físicament es genera una interrupció i el sistema operatiu s'encarrega de tenir-la disponible.

Quan la CPU esta en mode protegit, els programes de usuari tenen un accés limitat al joc de instruccions, i només el procés supervisor que estarà treballant en mode real estarà capacitat per realitzar determinades tasques. Això es així per evitar que els programes d'usuari puguin entrar en conflictes uns amb els altres tant en qüestió de memòria com de perifèrics. A més a més d'aquesta manera si un error de software penja un procés, els demés poden seguir treballant normalment i el sistema operatiu pot avortar el procés penjat. El sistema operatiu DOS no utilitza aquestes prestacions.

El 8086 així com tots els seus antecessors no disposen de cap mecanisme per ajudar a la multitasca ni a la memòria virtual des del processador, per el que es molt difícil realitzar un sistema operatiu multitasca per aquestes CPU. Òbviament el 80286 en mode protegit perd absolutament la compatibilitat amb els processadors anteriors.

De les característiques generals del 80286 podríem dir que te un bus de dades de 16 bits, un bus d'adreces de 24 bits amb el que pot adreçar fins 16Mbytes, te 25 instruccions més que el 8086 (ja implementades al 80188, 80186, NEC-V30, NEC-V20) i admet 8 modes d'adreçar. En mode protegit (virtual) permet

adreçar fins 1 Gigabyte. Les freqüències de treball van de 8 a 25 MHz. La instrucció més lenta utilitza 29 cicles de rellotge, de forma que un 80286 a 16 MHz. es capaç d'executar més de mig milió d'instruccions per segon, casi 15 vegades més que un 8086 a 8Mhz. amb el que es veu que no tot es el clock de la CPU, la optimització de l'execució de algunes instruccions es fonamental.

Posteriorment al 80286 el segueix el **80386**, que marca un abans i un després en els processadors de Intel. El 80386 disposa d'una arquitectura de registres de **32 bits**, amb un bus d'adreces de 32 bits amb el que pot adreçar fins 4 GB. (Gigabytes) de memòria física i més modes possibles de funcionament. El **mode real** que es compatible 8086, el **mode protegit** que es compatible amb el del 80286, un **mode protegit propi** que permet trencar la barrera dels tradicionals segments i el **mode virtual 86** en el que permet emular el funcionament simultani de varis 8086. Un cop mes, tots el modes son incompatibles entre si i requereixen un sistema operatiu específic. Amb el sistema operatiu DOS treballa en modes real, i no aprofita els 32 bits amb el que es desaprofita el 50% de rendiment. Aquestes CPU ja treballen amb un rellotge que va de 16 a 40 MHz. Es va fabricar una versió SX que tenia el bus de dades de 16 bits per mantenir la compatibilitat hardware amb el 80286, que necessitava 2 passades (accessos de bus) per accedir a les dades de 32 bits.
Es dona un pas més i es millora la velocitat dels processadors amb multiplicadors de rellotge interns i s'integra amb la CPU 80386 el coprocessador matemàtic 80387 i s'anomena a aquesta CPU 486, surt amb moltes versions, amb o sense coprocessador, amb 16 i 32 bits de bus, i evoluciona cap a unes freqüències de treball que van de 25 a 100Mhz. amb el epígrafs DX, DX2, DX4. En aquesta època molts fabricants CYRYX, IBM, AMD, etc. varen treure productes alternatius i compatibles, alguns amb algunes millores de velocitat i millor preu. Es important saber que algunes CPU han sortit amb petites errades d'execució en casos molt determinats d'algunes instruccions, els compiladors intenten no produir mai codi que inclogui aquestes excepcions.

El següent pas es més gran que l'anterior, Intel integra una CPU de 64 bits de bus de dades, es l'anomenat Pentium, amb el que els accessos a memòria dupliquen la seva velocitat i incorpora un molt alt nivell de optimització i una segmentació que permet, utilitzant compiladors amb un gran grau d'eficiència, executar simultàniament l'execució de dues instruccions consecutives. Esta proveït de dues memòries cache internes i te la capacitat de predir el destí dels salts i s'ha millorat el rendiment del coprocessador. La freqüència de treball en aquestes CPU va des de les primeres a 60 MHz. fins a les últimes a 233 MHz. A les darreres sèries es va incorporar el MMX que es un joc d'instruccions que fa més òptima la utilització de multimedi. AMD treu la sèrie K6 com a competència i la fa arribar a freqüències de treball de fins 550Mhz. i obté en algunes aplicacions un rendiment lleugerament més elevat. El Pentium en el fons son dos 486 al estil de Z80, que aprofiten els recursos per formar la CPU de 64 bits.

Posteriorment Pentium II, Pentium III, Pentium IV, i K6-II, K6-III, K7 de AMD…. que milloren l'accés a les caches internes, incorporació de més caché de segon

nivell, optimitzacions en l'execució, etc. en el moment d'escriure aquestes notes el K7 supera els 2000 MHz. i el Pentium IV a 2500Mhz.

Una característica important dels microprocessadors a partir del 80386 es la disponibilitat de memòries caché de alta velocitat d'accèss i que emmagatzema una petita part de la memòria principal. Quan la CPU accedeix a una posició de memòria, uns certs circuits de control s'encarreguen d'anar dipositant el contingut d'aquestes posicions i consecutives en memòria caché. Quan es necessari accedir a una instrucció o a una dada i ja es a la memòria caché, aquest accés es molt més ràpid. Lo ideal seria que tota la memòria fos tan ràpida com la caché, però per qüestió de preu no es viable. En un proper capítol veurem el funcionament d'aquestes memòries.

En la evolució de les CPU hi ha bàsicament un component tecnològic, tant la capacitat de integració, les tensions de treball cada vegada més baixes, les velocitats que es poden assolir, tot influeix en que de una CPU Pentium amb una amplada de pista de 0.80 micres i amb una superfície de 294mm2 amb 3,1 milions de transistors a un Pentiun IV fabricat amb pistes de 0,13 micres amb una superfície de 145mm2 i 55,0 milions de transistors, hi ha una evolució tecnològica important.

REGISTRES DEL 8086 I DEL 286.

Aquests processadors disposen de 14 registres de 16 bits (mes en el 286). La missió d'aquests registres es emmagatzemar dades i les posicions de memòria que experimentaran repetides manipulacions, ja que els accessos a memòria son molt més lents que els accessos als registres. Hi ha certes operacions que només es poden realitzar sobre els registres. No tots els registres serveixen per emmagatzemar dades, alguns estan especialitzats en apuntar a les adreces de memòria (apuntadors). La mecànica bàsica de funcionament d'un programa consisteix en carregar els registres amb dades de la memòria o d'un port de E/S, processar les dades i retornar el resultat a la memòria o a un altra port de E/S. Òbviament, si una dada només te d'experimentar un canvi, es preferible realitzar la operació directament sobre la memòria, si es possible. A continuació descriurem els registres del 8086.

AX	SP	CS	IP
BX	BP	DS	flags
CX	SI	SS	
DX	DI	ES	
Registres de dades	Registres apuntadors de la pila i índex	Registres de segment	Registre apuntador d'instruccions i flags

Registres de dades:

AX, BX, CX, DX: es poden utilitzar o be com a registres de 16 bits o com dos registres separats de 8 bits (bytes superior i inferior) canviant la X per H o L segons ens voléssim referir a la part alta o baixa respectivament. Per exemple, AX es descomposa en AH (part alta) i AL (part baixa). Evidentment, qualsevol canvi sobre AH o AL altera AX!: valgui com exemple que al incrementar AH s'estan afegint 256 unitats a AX.

AX = Acumulador.

Es el registre principal, es utilitzat en las instruccions de Multiplicació i divisió i en algunes instruccions aritmètiques especialitzades, així com en certes operacions de caràcter específic com entrada, sortida i traducció. Cal observar que el 8086 es suficientment potent per realitzar les operacions lògiques, la suma i la resta sobre qualsevol registre de dades, no necessàriament l'acumulador.

BX = Base.

S'utilitza com registre base per fer referència a adreces de memòria per amb adreçar de forma indirecte, mantenint l'adreça de la base o principi de taules o matrius. D'aquesta manera, no es necessari indicar una posició de memòria fixa, sinó la número BX (fent avançar de unitat en unitat a BX, per exemple, es pot anar accedint a un gran bloc de memòria en un bucle).

CX = Comptador.

S'utilitza normalment com comptador en bucles i operacions repetitives de manejament de cadenes. En les instruccions de desplaçament i de rotació s'utilitza com a comptador de 8 bits.

DX = Dades.

Utilitzat conjuntament amb AX en les operacions de multiplicació i divisió que generen dades de 32 bits. En les d'entrada i sortida se usa per especificar l'adreça del port E/S.

Registres de segment:

Defineixen àrees de 64 Kb dins del espai d'adreces de 1 Mb del 8086. Aquestes àrees poden solapar-se total o parcialment. No es possible accedir a una posició de memòria no definida per algun segment: si es necessari algun es tindrà de moure.

CS = Registre de segment de codi (**code segment**).

Conté l'adreça del segment amb les instruccions del programa. Els programes de més de 64 Kb requereixen canviar CS periòdicament.

DS = Registre de segment de dades (**data segment**).

Segment del àrea de dades del programa.

SS = Registre de segment de pila (**stack segment**).

Segment de pila.

ES = Registre de segment extra (**extra segment**).

Segment d'ampliació per la zona de dades. Es extraordinàriament útil actuant en conjunció amb DS: amb ambdós es poden definir dos zones de 64 Kb, tan allunyades com es vulgui en el espai d'adreces, entre las que es poden intercanviar dades.

Registres apuntadors de la pila:

SP = Apuntador de la pila (**stack pointer**).

Apunta al començament de la pila. Utilitzat en les instruccions de manejament de la pila.

BP = Apuntador base (base pointer).

Es un apuntador de base, que apunta a una zona dins de la pila dedicada al emmagatzematge de dades (variables locals i paràmetres de les funcions en els programes compilats).

Registres índex:

SI = Índex font (source índex).

Utilitzat com registre índex en certs modes d'adreçar de forma indirecte, també s'usa per guardar un valor de desplaçament en operacions de cadenes.

DI = Índex destí (destination índex).

S'usa en determinats modes d'adreçar de forma indirecte i per emmagatzemar un desplaçament en operacions amb cadenes.

Apuntador de instruccions o comptador de programa:

IP = Apuntador de instrucció (instruction pointer).

Marca el desplaçament de la instrucció en curs dins del segment de codi. Es automàticament modificat amb la lectura d'una instrucció.

Registre d'estat o de indicadors (flags).

Es un registre de 16 bits dels que 9 son utilitzats per indicar diverses situacions durant l'execució d'un programa. Els bits 0, 2, 4, 6, 7 i 11 son indicadors de condició, que reflecteixen els resultats de operacions del programa; els bits del 8 al 10 son indicadors de control i la resta no s'utilitzen. Aquests indicadors poden ser comprovats per les instruccions de salt condicional, el que permet variar el flux seqüencial del programa segons el resultat de les operacions.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	OF	DF	IF	TF	SF	ZF	-	AF	-	PF	-	CF

CF (Carry Flag): Indicador de carry. El seu valor correspon al que portem d'una suma o resta.

OF (Overflow Flag): Indicador de desbordament. Indica que el resultat de una operació no entra en la mida del operant destí.

ZF (Zero Flag): Indicador de resultat 0 o comparació igual.

SF (Sign Flag): Indicador de resultat o comparació negativa.

PF (Parity Flag): Indicador de paritat. S'activa després de l'execució de algunes operacions aritmètic-lògiques per indicar que el nombre de bits a 1 resultant es parell.

AF (Auxiliary Flag): Per ajust decimal a operacions BCD.

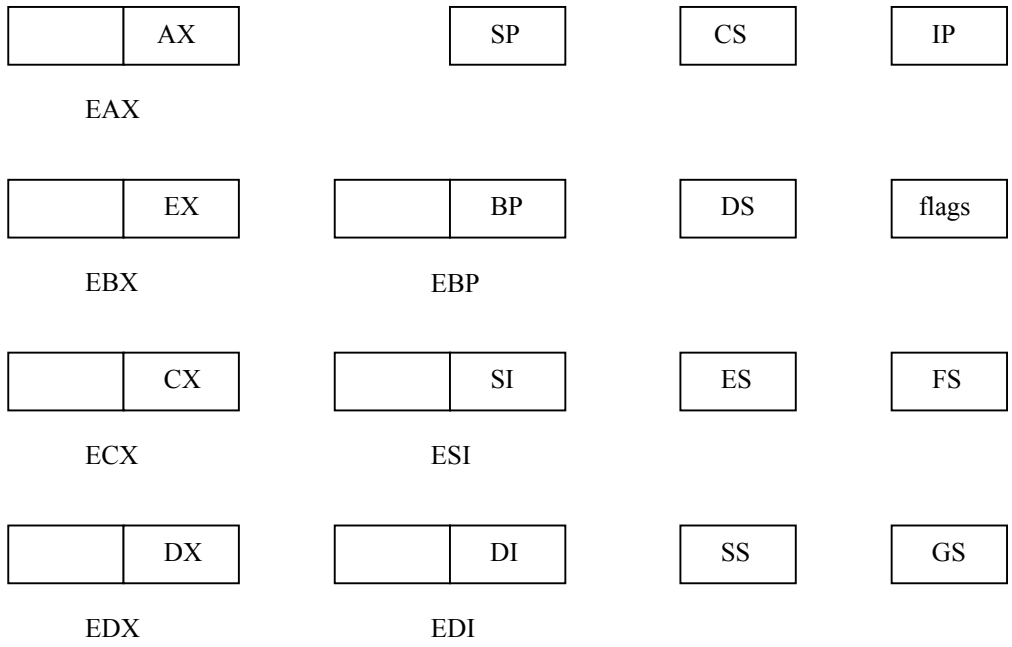
DF (Direction Flag): Indicador de direcció. Manipulant blocs de memòria, indica el sentit de avanç (ascendent/descendent).

IF (Interrupt Flag): Indicador d'interrupcions: quan esta a 1 estan permeses.

TF (Trap Flag): Indicador de trap (per permetre execució pas a pas).

REGISTRES DEL 386 I PROCESADORES SUPERIORS.

Els 386 i superiors disposen de molts més registres dels que anem a veure ara. Encara que sota el sistema operatiu DOS solament es solen emprar els que veurem, que constitueixen bàsicament una extensió a 32 bits dels registres originals del 8086.



S'amplia la mida dels registres de dades (que poden ser accedits en fragments de 8, 16 ó 32 bits) i s'afegeixen dos nous registres de segment de propòsit general (FS y GS). Alguns dels registres que veurem aquí son realment de 32 bits (com EIP en lloc de IP), però sota el sistema operatiu DOS no poden ser emprats de manera directa, per tant no els considerarem.

MODES D'ADREÇAMENT.

Son les diverses maneres d'accedir a les dades a memòria per part del processador. Abans de veure els modes d'adreçar, podem veure com es la sintaxi general de les instruccions, ja que en posarem alguna en els exemples:

INSTRUCCIÓ DESTI, FONT

On destí indica on es deixa el resultat de la operació en la que poden participar (segons casos) FONT i el propi DESTI. Hi ha instruccions, que solament tenen un operant, com la següent, o fins i tot cap:

INSTRUCCIÓ DESTI

Com exemples, encara que no les hem vist, n'utilitzarem un parell: la de copia o moviment de dades (MOV) i la de suma (ADD).

ORGANITZACIÓ D'ADRECES: SEGMENTACIÓ.

Com ja sabem, els microprocessadores 8086 i compatibles tenen registres de una mida màxima de 16 bits que adreçarien fins 64K; en canvi, la adreça es composa de 20 bits amb capacitat per 1Mb, haurem per tant de recorre a algun artifici para adreçar tota la memòria. Aquest artifici consisteix en la segmentació: es tracta de dividir la memòria en grups de 64K.
Cada grup s'associa amb un registre de segment; el desplaçament (offset) dintre d'aquest segment el proporciona un altre registre de 16 bits. L'adreça absoluta es calcula multiplicant per 16 el valor del registre de segment i sumant l'offset, obtenint una adreça efectiva de 20 bits. Això es equivalent a concebre el mecanisme de generació de l'adreça absoluta, com si es tractes de que els registres de segment tinguessin 4 bits a 0 (imaginaris) a la dreta abans de sumar el desplaçament:

adreça = segment * 16 + offset

A la pràctica, una adreça se indica amb la notació SEGMENT:OFFSET; a més a mes, una mateixa adreça es pot expressar de més d'una manera: per exemple, 3D00h:0300h es equivalent a 3D30:0000h. Es important ressaltar que no es pot accedir a més de 64 Kb en un segment de dades. Por això, en els processadors 386 i superiors s'han d'emprar registres de 32 bit per generar adreces (sota DOS), encara que per els càlculs poden ser interessants (no obstant, sí seria possible configurar aquests processadors para poder adreçar mes memòria sota DOS amb els registres de 32 bits, encara que no resulta generalment gens pràctic).

TIPUS DE MODES D'ADREÇAT.

Adreçat en mode immediat: El operant es una constant situada darrera del codi de la instrucció. De totes maneres, com registre destí no es pot indicar un registre de segment (es tindrà de utilitzar un registre de dades com a pas intermedi).

```
ADD   AX,0fffh
```

El número hexadecimal 0fffh es la constant numèrica que en el adreçat immediat se sumarà al registre AX. Al treballar amb assembladors, es poden definir símbols constants (no variables) i es més intuïtiu:

```
dada   EQU   0fffh            ; símbol constant
       MOV   AX,dada
```

Si se referència a l'adreça de memòria de una variable de la següent forma, també es tracta de un cas d'adreçat immediat:

```
dada    DW    0fffh              ; ara es una variable
        MOV    AX,OFFSET dada    ; AX = "Adreça de
                                 ; memòria" de dada
```

Perquè s'ha de tenir compta que quan traduïm a números el símbol podria quedar:

```
17F3:0A11    DW    FFF
             MOV   AX,0A11
```

Adreçat de registre: Els operants, necessàriament de la mateixa mida, estan continguts en els registres indicats en la instrucció:

```
MOV   DX,AX
MOV   AH,AL
```

Adreçat directe o absolut: L'operat esta situat a l'adreça indicada a la instrucció, relativa al segment que es tracti:

```
MOV   AX,[57D1h]
MOV   AX,ES:[429Ch]
```

Aquesta sintaxi (sense la 'h' de hexadecimal) seria la que admet el programa DEBUG (realment s'hauria de posar, en el segon cas, ES: en una línia i el MOV en una altra). Al treballar amb assembladors, les variables en memòria es poden fer referència amb etiquetes simbòliques:

```
MOV   AX,dada
MOV   AX,ES:dada

dada    DW    1234h       ; variable del programa
```

En el primer exemple es transfereix a AX el valor contingut a l'adreça apuntada per l'etiqueta dada sobre el segment de dades (DS) que s'assumeix per defecte; en el segon exemple s'indica de forma explícita el segment dons es tracta del segment ES. L'adreça efectiva es calcula de la forma ja vista amb anterioritat:

> Registre de segment * 16 + desplaçament_de_dada
> (Aquest desplaçament depèn de la posició al assemblar el programa).

Adreçat indirecte: L'operant es troba en una adreça assenyalada per un registre de segment*16 més un registre base (BX/BP) o índex (SI/DI). (Nota: BP actua per defecte amb SS).

```
MOV   AX,[BP]              ; AX = [SS*16+BP]
MOV   ES:[DI],AX           ; [ES*16+DI] = AX
```

Indirecte amb índex o indexat: El operant es troba en una adreça determinada per la suma d'un registre de segment*16, un registre índex, SI o DI y un desplaçament de 8 ó 16 bits. Exemples:

```
MOV   AX,[DI+DESP]      ó      MOV   AX,desp[DI]
ADD   [SI+DESP],BX      ó      ADD   desp[SI],BX
```

Indirecte amb base i índex o indexat a base: El operant es troba adreça especificada per la suma d'un registre de segment*16, un de base, un de índex i opcionalment un desplaçament de 8 ó 16 bits:

```
MOV   AX,ES:[BX+DI+DESP]  ó  MOV   AX,ES:desp[BX][DI]
MOV   CS:[BX+SI+DESP],CX  ó  MOV   CS:desp[BX][SI],CX
```

Combinacions de registres de segment i desplaçament.

Com es veu en els modes d'adreçat, hi ha casos en els que s'indica explícitament el registre de segment a utilitzar per accedir a les dades. Existeixen uns segments associats per defecte als registres de desplaçament (IP, SP, BP, BX, DI, SI); solament es necessari declarar el segment quan no coincideix amb l'assignat per defecte. En aquest cas, l'assemblador genera un byte addicional (a modes de prefix) per indicar quin es el segment al que es fa referència. La següent taula relaciona les possibles combinacions dels registres de segment i els de desplaçament:

	CS	SS	DS	ES
IP	SI	NO	NO	NO
SP	NO	SI	NO	NO
BP	AMB PREFIX	PER DEFECTE	AMB PREFIX	AMB PREFIX
BX	AMB PREFIX	AMB PREFIX	PER DEFECTE	AMB PREFIX
SI	AMB PREFIX	AMB PREFIX	PER DEFECTE	AMB PREFIX
DI	AMB PREFIX	AMB PREFIX	PER DEFECTE	AMB PREFIX(1)

(1) També per defecte en el manejament de cadenes.

Els 386 i superiors admeten altres modes d'adreçat més sofisticats, que veurem més endavant, després de conèixer totes les instruccions del 8086. Per ara, amb tots aquests modes que hem vist podem considerar que n'hi ha més que suficient. De fet, alguns s'utilitzen en molt comptades ocasions.

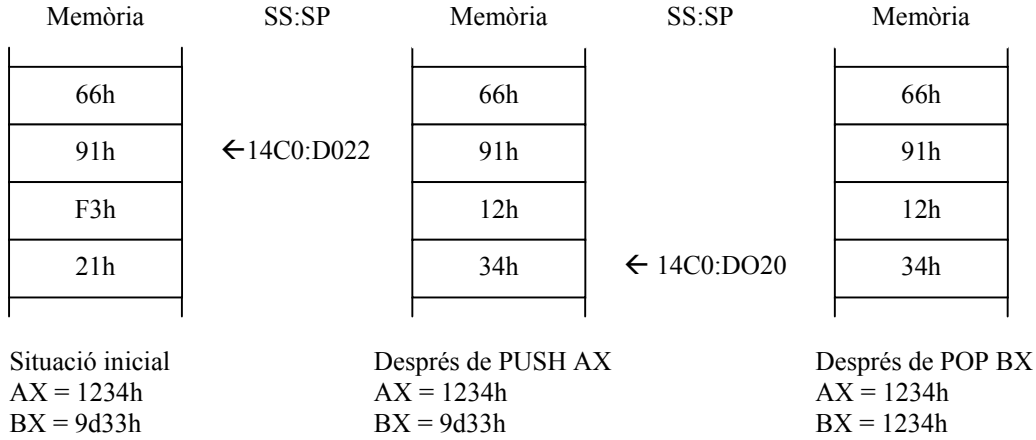
LA PILA.

La pila es un bloc de memòria de estructura LIFO (Last Input First Output: últim d'entrar, primer de sortir) que s'adreça mitjançant desplaçaments des del registre SS (segment de pila). Les posicions individuals dintre de la pila

es calculen sumant al contingut del segment de pila SS un desplaçament contingut en el registre apuntador de la pila SP. Totes les dades que s'emmagatzemen a la pila son de longitud paraula, i cada vegada que s'hi introdueix quelcom mitjançant les instruccions de manejament de la pila (PUSH i POP), l'apuntador decreix en dos; es a dir, la pila avança cap adreces decreixents. El registre BP sol utilitzar-se normalment per apuntar a una certa posició de la pila i accedir mitjançant un índex als seus elements (generalment en el cas de variables) sense necessitat de treure'ls de la pila per consultar-los.

La pila es utilitzada freqüentment al principi d'una subrutina per preservar els registres que no es desitgen modificar; al final de la subrutina n'hi ha prou amb recuperar-los en ordre invers al que foren dipositats. En aquestes operacions convé tenir cura, ja que la pila en els 8086 es comú al processador i al usuari, per tan el que se emmagatzema en aquesta també les adreces de retorn de les subrutines. Aquesta última es, de fet, la més important de les seves funcions. La estructura de pila permet que unes subrutines cridin a altres que a la vegada puguin cridar a altres i així successivament: a la pila s'emmagatzemen les adreces de retorn, que seran les de la següent instrucció que va provocar la crida a la subrutina. Així, al retornar de la subrutina se extreu de la pila l'adreça a on tornar. Els compiladors dels llenguatges d'alt nivell la utilitzen també per passar els paràmetres dels procediments i per generar en aquesta les variables automàtiques - variables locals que existeixen durant la execució del subprograma i es destrueixen immediatament després -. Per això, una norma bàsica es que es te que de treure de la pila sempre tot el que s'ha apilat per evitar una pèrdua de control immediata de l'ordinador.

Exemple de operació sobre la pila (totes les dades son arbitraries):



UN PROGRAMA D'EXEMPLE.

Encara que les instruccions del processador no seran vistes fins una mica més endavant, amb objecte d'ajudar a la imaginació de l'alumne elaborarem un primer programa de exemple en llenguatge assemblador. La utilitat de aquest programa es deixar patent que l'únic que entén el 8086 son números, encara

que nosaltres ens referirem a ells amb uns símbols que faciliten entendre'ls. També es interessant aquest exemple per fiançar el concepte de registre de segment.

En aquest programa solament utilitzarem les instruccions MOV, ja coneguda, i alguna altra més com la instrucció INC (incrementar), DEC (disminuir una unitat) i JNZ (saltar si el resultat no es zero). Suposem que el programa esta ubicat a partir de la posició de memòria 14D3:7A10 (arbitràriament escollida) i que el que pretenem fer amb ell es netejar la pantalla. Com que l'ordinador es un PC amb monitor de color, la pantalla de text comença en B800:0000 (no es més que una zona de memòria). Per cada caràcter que hi ha en aquesta pantalla, començant a dalt a l'esquerra, a partir de l'adreça B800:0000 tenim dos bytes: el primer, amb el codi ASCII del caràcter i el segon amb el color. El que anem a fer es omplir els 2000 caràcters (80 columnes x 25 línies) amb espais en blanc (codi ASCII 32, ó 20h en hexadecimal), sense modificar el color que hi havia abans. Això es, es tracta de posar el valor 32 en l'adreça B800:0000, la B800:0002, la B800:0004... i així successivament.

El programa romandria a memòria d'aquesta manera: La primera columna indica l'adreça de memòria on esta el programa que se executa (CS=14D3h i IP=7A10h al principi). La segona columna es el codi maquina que interpreta el 8086. Algunes instruccions ocupen un byte de memòria, altres dos tres (n'hi ha de mes). La tercera columna conte el nom de les instruccions, quelcom molt més llegible per els humans que els números:

```
14D3:7A10      B9 D0 07       MOV  CX,7D0H   ; CX = 7D0h (2000 decimal = 7D0 hexadecimal)
14D3:7A13      B8 00 B8       MOV  AX,0B800h ; segment de la memòria de pantalla
14D3:7A16      8E D8          MOV  DS,AX     ; apuntar segment de dades a la misma
14D3:7A18      BB 00 00       MOV  BX,0      ; apuntar al primer caràcter ASCII
                                             ; de la pantalla
14D3:7A1B      C6 07 20       MOV  BYTE PTR [BX],32
                                             ; es posa BYTE PTR per indicar que 32
                                             ; es de 8 bits
14D3:7A1E      43             INC  BX        ; BX=BX+1 -< apuntar al byte de color
14D3:7A1F      43             INC  BX        ; BX=BX+1 -< apuntar al següent
                                             ; caràcter ASCII
14D3:7A20      49             DEC  CX        ; CX=CX-1 -< queda un caràcter menys
14D3:7A21      75 F8          JNZ  -8        ; si CX no es 0, saltar 8 bytes enrere
                                             ; (a 14D3:7A1B)
```

Com es pot veure, la segona instrucció (bytes de codi maquina 0B8h, 000h i 0B8h posats en posicions consecutives) esta col·locada a partir del desplaçament 7A13h, ja que l'anterior que ocupava 3 bytes començava en 7A10h.

En el exemple carreguem el valor 0B800h a DS amb l'ajut de AX com intermediari. El motiu es que els registres de segment no admeten l'adreçat immediat. A mida que es van fent programes, l'assemblador dona missatges d'error quan es troba aquestes errades i permet anar aprenent amb facilitat les normes, que tampoc son masses. La instrucció MOV BYTE PTR [BX],32 equival a dir: «posar a la posició de memòria apuntada per BX (DS:[BX] per ser més exactes) el byte de valor 32». El valor 0F8h del codi maquina de la última instrucció es el complement a dos (número negatiu) del valor 8.

Normalment, casi mai tindrem d'assemblar a ma consultant unes taules, com hem fet en aquest exemple. De totes formes, la millor manera d'aprendre assemblador es no oblidant l'estreta relació de cada línia de programa amb la CPU i la memòria.