1 Famiglia 80X86

In questo capitolo facciamo una panoramica delle principali caratteristiche delle CPU della famiglia X86 successive all'8086.
Alcune delle caratteristiche più interessanti delle CPU avanzate della serie X86 richiedono una conoscenza dei concetti fondamentali dei S.O. , che illustreremo nella altrove.

8086 e 8088

La CPU capostipite della famiglia X86 è l'8086, progettato per essere parzialmente compatibile, a livello di codice sorgente, con una precedente CPU di Intel ad 8 bit, l'8080.
L'8086 aveva una struttura di registri simile a quella dell'8088. Nonostante i codici mnemonici delle istruzioni fossero parzialmente compatibili, i codici operativi della nuova CPU non erano uguali a quelli dell'8080, quindi i programmi per 8086 richiedevano almeno di essere ricompilati.
Si ricorda inoltre (vedi Volume 1) che l'unica differenza significativa fra le 8086 e 8088 è costituita dal fatto che l'8086 aveva un bus dati da 16 bit, mentre l'8088, pur avendo anch'esso un parallelismo interno di 16 bit, aveva un bus dati di soli 8 bit.

8087

Le CPU della famiglia 80X86 nacquero con strumenti automatici per l'integrazione di un coprocessore aritmetico. Un coprocessore è un chip esterno il cui hardware è dedicato alle sole operazioni aritmetiche.
La particolarità interessante è che le istruzioni per il coprocessore fanno parte del normale programma per 8086. L'8086 provvede quindi normalmente alla fase di fetch e, quando si accorge che l'istruzione da eseguire non è "la sua", ne passa il codice operativo al coprocessore e si mette in attesa che essa venga completata*. L'8087 mette a disposizione istruzioni per l'aritmetica a virgola mobile e per il calcolo di funzioni trascendenti e logaritmiche.

80186

Questa CPU fu introdotta quasi contemporaneamente alla più potente 80286, che guadagnò da subito il favore del mercato dei personal computer.
È un'estensione dell'8086, con istruzioni ottimizzate e più potenti, con molte delle caratteristiche del 286, ma senza le due più innovative: modo protetto e memoria virtuale.
Commercialmente non ebbe fortuna come processore general purpose, ma fu ed è tuttora utilizzato come microcontrollore per applicazioni "embedded". Esiste anche una CPU 80188, con bus dati di 8 bit, invece di quello da 16 bit dell'80186.

80286

Rispetto all'8086 non cambia il parallelismo interno, né quello del data bus. Aumenta fino a 16 MByte la memoria indirizzabile (24 bit d'indirizzo).
Nelle funzionalità del microprocessore sono invece presenti cambiamenti fondamentali. Si introducono funzioni utili nella realizzazione di sistemi operativi multitasking. Esse permettono la protezione della memoria dei singoli task (detti anche processi), cioè l'indicazione di errore nel caso che un task tenti di avere accesso a zone di memoria ad esso non riservate. Oltre alla protezione della memoria, il 286 ha un meccanismo per aiutare il Sistema Operativo nella realizzazione della memoria virtuale.
L'80286 può lavorare in due modalità molto diverse: modo **reale** (real address mode) e modo **protetto** (protected mode).
Se un 80286 lavora in modo reale esso è completamente compatibile con un 8086, funziona come un 8086 più veloce, anche l'indirizzamento della memoria è a 20 bit.
Se invece un 286 funzione in modo protetto esso avrà protezione della memoria e delle istruzioni e visibilità della memoria virtuale segmentata.
In modalità protetta i registri di segmento vengono trattati in modo diverso e non sono più sovrapposti all'offset, ma "affiancati" (vedi Capitolo XXXX). L'offset è ancora di 16 bit, come nell'886, per cui la dimensione massima dei segmenti è di 64 kByte.
Il S.O. MS DOS è stato scritto per funzionare in modo reale. I S.O. più moderni (Win95, OS/2, Linux) funzionano invece in modo protetto (anche se nella modalità protetta del 386, che è in piuttosto diversa).

Registri 286

Essendo la CPU a 16 bit, come l'8086, non ci sono cambiamenti, i registri che erano già presenti nell'8086 rimangono uguali.
Sono aggiunti alcuni registri di sistema per l'utilizzazione della memoria virtuale e la realizzazione del multitasking; sono inoltre aggiunti dei registri speciali per la gestione della modalità protetta.
Questi registri verranno analizzati, insieme al dettaglio del funzionamento della modalità protetta, in altra parte di questo volume.

* In verità una volta comunicata l'istruzione alla FPU, la CPU si mette ad eseguire la successiva. Se il programmatore vuole che la CPU si fermi ad attendere il risultato dell'operazione a virgola mobile deve usare un'istruzione FWAIT o WAIT.

80287

Come per l'8086, che aveva un 8087, il 286 ha un coprocessore aritmetico, il 287. In verità ce ne furono diversi e nacque una industria di terze parti che fornivano coprocessori più potenti del 287.
Come per il 286, anche il suo coprocessore matematico non aveva possibilità di tornare in modo reale una volta entrati in modalità protetta

80386

L'80386 ha un parallelismo interno di 32 bit, per cui può lavorare con registri da 32 bit. Anche il parallelismo del data bus è di 32 bit, per cui la massima quantità di memoria fisica indirizzabile da un 386 è di 4 GByte.
Come il 286 anche il 386 può lavorare in **modo reale** o in modo **protetto**, in più è presente una terza modalità di funzionamento: il **modo 8086 virtuale** (V86).
Nel caso della modalità reale il 386 si comporta come un 8086, con lo stesso modo di accesso segmentato alla memoria, a 20 bit di indirizzo (1 MByte), e con lo stesso tipo di interrupt vettorizzato.
Il vantaggio notevole, presente anche in modo reale, è che sono disponibili i registri e le istruzioni a 32 bit e diverse istruzioni nuove o potenziate. Sono inoltre presenti due nuovi registri di segmento.
Nel caso dell'uso della modalità protetta il 386 è una vera CPU a 32 bit per sistemi operativi multitasking, con supporto per la memoria virtuale sia segmentata che paginata, e protezione degli accessi e delle istruzioni.
Infine se la CPU funziona in modo V86 vengono mantenuti attivi i meccanismi di protezione e virtualizzazione della memoria tipici del modo protetto, ma la CPU funzionare come se fosse un 8086, con lo stesso set d'istruzioni, in parte estese a 32 bit, e con le stesse modalità di accesso alla memoria (la segmentazione "reale" di triste memoria). La modalità V86 serve perciò per far girare in modo protetto il software scritto per MS-DOS su Sistemi Operativi multitasking.
La memoria virtuale del 386 può avere un modello segmentato, come quella del 286, ed un nuovo modello a domanda di pagina, o "paginato". Il programmatore di sistema può scegliere se utilizzare per la memoria virtuale entrambi i modi, uno solo o di non avere memoria virtuale. Nei sistemi operativi multitasking più importanti presenti oggi sul mercato viene utilizzata solo la tecnica della paginazione.
Le ultime versioni del 386 avevano la possibilità di mettersi in modo di funzionamento a consumo ridotto ("rallentamento" del clock o sospensione di tutte le attività). Per gestire questa modalità, presente anche nelle CPU successive, sono state introdotte alcune nuove istruzioni di macchina.

Registri 386

L'ALU della CPU 386 è a 32 bit, per cui i tutti i registri sono stati espansi fino a 32 bit.
Sono stati introdotti i registri generali EAX, EBX, ECX, EDX (extended AX, ..) di 32 bit, suddivisibili in AX, BX, .. di 16 bit e AH, AL, .. di 8 bit. Analogamente sono stati definiti i registri EBP, ESI, EDI, ESP, di 32 bit, dei quali fanno parte i registri a 16 bit BP, SI, DI, SP, il loro funzionamento è del tutto analogo al caso a 16 bit. Analogamente al caso 8086 gli accessi in memoria che utilizzano EBP funzionano nel segmento di stack.
<FILE>
!!!! da fare
</FILE>

Figura 1: principali registri 80386

I registri di segmento CS, DS, SS e ES, di 16 bit, hanno le stesse funzioni dei rispettivi registri 8086. In modalità reale funzionano come normali registri di segmento 8086, in modalità protetta come "selettori di segmento" per la memoria virtuale (vedi Capitolo XXXX).
Sono introdotti i nuovi registri di segmento FS e GS, a disposizione del programmatore e non usati in modo implicito da nessuna istruzione, essi sono registri di segmento "in più", da tenere a disposizione, in modo analogo al registro ES nell'8086. I due nuovi registri di segmento si possono usare anche in modo reale (naturalmente i questo caso funzionano come se fossero su un 8086).
L'instruction pointer è a 32 bit e si chiama, ovviamente, EIP. Quando un 386 lavora in modalità reale utilizza solo la parte bassa di questo registro (IP, a 16 bit).
Sono presenti i registri, che erano anche nel 286, per l'accesso alle tabelle dei descrizione della memoria virtuale (GDTR e LDTR), naturalmente in questo caso essi devono essere più grandi. Infatti, visto che gli indirizzi di memoria fisica sono a 32 bit, la parte "inizio del segmento" di questi registri deve essere a 32 bit, mentre la parte "lunghezza del segmento" è di 20 bit; la sua utilizzazione viene descritta in seguito.
Esistono dei nuovi "registri di controllo", usati per utilizzare le nuove del modo protetto. Questi registri sono: CR0,CR1,CR2 e CR3.

Accesso alla memoria: calcolo dell'indirizzo e indirizzamento scalato

Trascurando per il momento il funzionamento della parte di segmento dell'indirizzo, che verrà sviluppata in altro Capitolo, consideriamo solo la parte di offset. Nel 386 l'offset negli accessi alla memoria in modo protetto può essere di 32 bit, per cui i segmenti possono essere lunghi fino a 4 GByte.
Nel 386 è stato introdotto un interessante modo di calcolare l'indice di una struttura dati (detto "**Scaled** Index Base"), o indirizzamento "**scalato**".
Con l'indirizzamento scalato è possibile moltiplicare, a tempo d'esecuzione, l'indirizzo di indice che si utilizza (p. es. EDI o EBX) per 1, 2, 4 o 8 (in realtà si tratta di una shift!).

In questo modo si può far coincidere l'indice della struttura dati con il contenuto del registro indice, rendendo così programmi più chiari e quindi più affidabili.
In sintesi, l'offset complessivo può essere formato da:

[BASE + INDICE * SCALA + DISPLACEMENT]

A differenza dell'8086, TUTTI i registri a 32 bit (generali (EAX, . .) e "puntatori" (ESI, . .)), possono essere usati sia come BASE che come INDICE. Il valore di SCALA può essere solo: 1, 2, 4, o 8. Il DISPLACEMENT può non esserci, essere a 16 bit oppure a 32 bit.
Esempi:

```
<CODICE>
    MOV EBX, [EDX + ESI * 4 + 6]
    MOV EBP, [EAX + EAX * 2] ; l'offset è EAX * 3
                             ; di improbabile utilità :-(
                             ; ma si può fare ..
</CODICE>
```

Questo nuovo modo d'indirizzamento può essere usato anche in real mode 386.

Particolari sui set d'istruzioni

186 e 286
Alcune istruzioni già presenti nell'8086 sono state rese più flessibili.
Si possono usare PUSH e POP con operandi immediati:

```
P. es.      PUSH 20 ; mette il numero 20 in cima allo stack
```

Le istruzioni di spostamento dei bit possono essere usate con numero di spostamenti in immediato:

```
P. es.      SHL AX, 3   ; esegue uno shift a sinistra di tre posti sul registro AX
```

Sono introdotte nuove istruzioni per il modo protetto (CLTS, LGDD, LLPP, LIDT, LMSW, LTR).

Una limitazione del 286, che ha molto limitato la portata dei sistemi operativi multiprogrammati scritti per questa CPU, è che non esiste un meccanismo affidabile per uscire dal modo protetto e tornare in modo reale. Una volta entrati in modo protetto, non è possibile tornare in modo reale.

PUSHA, POPA: salvataggio dei registri generali (186>)
PUSHA salva nello stack il contenuto di tutti gli 8 registri generali. Per riprenderli, usare POPA

INS, OUTS: I/O di stringa (186>)
IN e OUT "di stringa": INS trasferisce in ES:[DI] ciò che legge al port di I/O il cui indirizzo è in DX; poi incrementa automaticamente DI (o EDI nel 386), se Direction flag = 0, oppure lo decrementa, se Direction =1.
OUTS trasferisce il byte letto in DS:[SI] al port il cui indirizzo è in DX, poi cambia SI, in base al valore del Direction flag.
INS e OUTS, hanno le versioni a byte, word e double word: INSB, INSW, INSD; OUTSB, OUTSW, OUTSD, e come le altre istruzioni di stringa, possono essere preceduti da REP, REPE, REPNE (vedi il Volume 1 per i dettagli sulle istruzioni di stringa).

BOUND: controllo dei limiti di un array (186>)
Verifica se l'intero con segno specificato rientra all'interno di limiti specificati. Se ciò non accade si genera l'eccezione INT 5. Utile per verificare "automaticamente" se si esce fuori da un array.

```
BOUND <Registro>, <Limiti dell'array>
; funzionamento:
; INT # 5 se   (<Registro> < limite inferiore) OR
               (<Registro> > limite superiore) OR
```

<Registro> può essere da 16 o da 32 bit, è l'indice dell'array che si vuole controllare.
<Limiti dell'array> sono due numeri con segno in memoria, in locazioni consecutive, di 16 o di 32 bit a seconda del registro usato, che esprimono il valore minimo e massimo dell'indice dell'array.

Esempi:

```
..
; definizione di un array con indici da 0 a 100:
MOV [LimitiArray], 0
MOV [LimitiArray + 2], 100
..
; controllo dei limiti:
```

```
    BOUND SI, [LimitiArray]
    ; se arrivo qui non è scattato l'INT 5, allora l'indice (SI) è far 0 e 100
    ; suppongo che l'array sia fatto di double word e carico l'elemento
    ; di indice SI:
    MOV EAX, [Array + SI * 4]
```

ENTER e LEAVE: "stack frame" (186>)
ENTER e LEAVE provvedono alla creazione di un "ambiente di stack" (stack frame), cioè uno spazio nello stack ove la procedura può allocare le sue variabili locali, che saranno distrutte prima della RET. In modo reale funzione a 16 bit (con BP ed SP), in modo protetto a 32 bit (con EBP ed ESP).
Sintassi:

```
    ENTER <Dimensione Stack frame>, <Immediato a 8 bit>
    ; prepara lo spazio per le variabili locali di una procedura
    ; funzionamento (con <Immediato a 8 bit> = 0):
    : "PUSH EBP"
    ; "EBP <- ESP"
    ; "ESP <- ESP - <Dimensione Stack frame>
    LEAVE
    ; distrugge le variabili locali di una procedura
    ; ripristinando i valori di ESP e EBP
    ; funzionamento:
    ; "ESP <- EBP"
    ; "POP EBP"
```

<Dimensione Stack frame> è un numero in immediato di 16 bit che specifica quanti byte devono essere occupati nello stack per le variabili locali della procedura.
<Immediato a 8 bit> è un numero da 0 a 31 che indica il "livello di innestamento" (nesting level), che qui trascureremo, considerandolo 0.
ENTER e LEAVE sono tipicamente la prima e l'ultima delle istruzioni di una procedura.
ENTER crea lo spazio voluto nello stack, memorizzando EBP, copiandovi il valore corrente di ESP, e spostando lo stack pointer. Dunque dopo la ENTER EBP punta ai parametri della procedura, mentre ESP punta all'inizio delle variabili locali e può essere anche usato per le operazioni locali sullo stack.
LEAVE ripristina le condizioni di EBP ed ESP prima della ENTER, cancellando di fatto le variabili locali e preparando la RET, che seguirà subito dopo.

```
    ..
Routine PROC
    ENTER 5, 0   ; Definisce uno stack frame di 5 byte
                 ; usati da una variabile boolean
                 ; e da un integer da 32 bit
    MOV BYTE PTR [ESP], 0    ; inizializza a false la variabile
                             ; locale boolean
    ; inizializza a -1 la variabile locale integer da 32 bit:
    MOV BYTE PTR [ESP + 1], 0FFFFFFFFh

    MOV EAX, [EBP]   ; legge dallo stack il valore del primo
                     ; parametro passato dal programma chiamante
    ..  ; esegue tutta la procedura
    LEAVE   ; ripristina EBP ed ESP per la RET
    RET
Routine ENDP
        ..
```

386
Il set d'istruzioni 386, oltre a comprendere l'estensione a 32 bit di tutte le istruzioni che prima lavoravano a 16 bit, include anche diverse nuove istruzioni.
Inoltre, a differenza delle CPU precedenti, i flag del 386 sono contenuti in un registro a 32 bit (EFLAGS, Enhanced FLAGS)). Sono introdotti diversi nuovi flag, che hanno a che fare con le nuove modalità di funzionamento in modo protetto e V86.
Per esempio alcuni flag nuovi, sono:
VM ("**V**irtual **M**achine flag"), individua se la CPU sta funzionando in modalità virtuale 8086 (V86) o no.
EM ("**Em**ulate") dice se la CPU deve emulare il set d'istruzioni del coprocessore matematico
PE ("**P**rotection **E**nabled") a 1 se la CPU sta funzionando in modo protetto
RF ("**R**esume **F**lag") esclude temporaneamente la generazione delle eccezioni, ciò per far funzionare regolarmente il debugger.

Una nota riguardo al flag di interrupt. In modo protetto e V86 è possibile modificare il flag di interrupt solo con le istruzioni CLI e STI, che possono essere "protette", possono perciò essere eseguite solo dai programmi che ne hanno il diritto. Per ragioni di sicurezza non è possibile modificare l'interrupt flag eseguendo delle POPF dallo stack.

MUL, IMUL: operando a 8, 16 o 32 bit
Se l'operando è a 8 o 16 bit, l'operazione è la stessa dell'8086; se è a 32 bit è una estensione "logica": il risultato è a 64 bit e finisce nella coppia di registri EDX, EAX; in EDX la parte alta. Analogo al caso 8086 il comportamento del flag (vedi Volume 1).

DIV, IDIV: operando a 8, 16 o 32 bit
Se l'operando è a 32 bit il dividendo è nella coppia di registri EDX, EAX, il quoziente finisce in EAX, il resto in EDX. Errori nell'operazione lanciano l'eccezione INT 0.

PUSHFD e POPFD: salvataggio flag estesi (386>)
Nuove istruzioni, senza operandi, che usano lo stack per la memorizzazione del registro dei flag a 32 bit (EFLAGS). Naturalmente le PUSH e POP normali nel 386 ammettono operandi da 32 bit, come p.es. EAX.

PUSHAD e POPAD: salvataggio dei registri generali a 32 bit (386>)
Analoghe a double word della PUSHA e della POPA, salvano nello stack e, rispettivamente, ripristinano TUTTI gli otto registri generali a 32 bit della CPU (nell'ordine di inserzione nello stack: EAX, ECX, EDX, EBX, il valore di ESP prima che EAX fosse messo nello stack, EBP, ESI, EDI). Lo stack "cresce" verso indirizzi più bassi, così come nell'8086. Le due istruzioni hanno solo un operando.

Operazioni di "**bit scan** e **bit test**" (386>)
Le istruzioni di bit scan (BSF e BSR) cercano il primo bit diverso da zero nell'operando sorgente e ne indicano il numero nell'operando destinazione.
Le istruzioni di bit test hanno due operandi uno dei quali indica la destinazione su cui si lavora e l'altro un bit specifico della destinazione, su cui fare diverse operazioni. Prima di modificare il bit specificato la CPU nel copia il valore nel flag di carry. Nelle istruzioni di bit test lo stato dei flag OF, SF, ZF, AF e PF è indeterminato. Queste istruzioni sono: BT, BTC, BTR, BTS.

```
BSF  <RegistroDestinazione>, <Sorgente>
; Bit Scan Forward
; funzionamento:
; ZF <- 1 se <Sorgente> =0, 1 se c'è almeno un bit a 1
; <OperandoDestinazione> <- numero del primo bit a 1 dal meno significativo

BSR  <RegistroDestinazione>, <Sorgente>
; Bit Scan Reverse
; funzionamento:
; ZF <- 1 se <Sorgente> =0, 1 se c'è almeno un bit a 1
; <OperandoDestinazione> <- numero del primo bit a 1 dal più significativo

BT   <Op1>, <Op2>
; Bit Test
; copia nel Carry Flag il bit di <Op1> indicato da <Op2>
; funzionamento:
; CF <- bit di <Op1> il cui numero è <Op2>
; <Op2> può essere un registro od un op. in immediato (non memoria)

BTC <Destinazione>, <Op2>
; Bit Test and Complement
; copia nel Carry Flag il valore del bit di <Destinazione> indicato da <Op2>
; ed inverte il valore dello stesso bit in <Destinazione>
; funzionamento:
; CF <- bit di <Destinazione> il cui numero è <Op2>
; <Destinazione> <- <Destinazione>, ma con il bit numero <Op2> invertito

BTR <Destinazione>, <Op2>
; Bit Test and Reset
; copia nel Carry Flag il valore del bit di <Destinazione> indicato da <Op2>
; e mette a zero il valore dello stesso bit in <Destinazione>
; funzionamento:
; CF <- bit di <Destinazione> il cui numero è <Op2>
; <Destinazione> <- <Destinazione>, ma con il bit numero <Op2> = 0

BTS <Destinazione>, <Op2>
; Bit Test and Set
; copia nel Carry Flag il valore del bit di <Destinazione> indicato da <Op2>
; e mette a uno il valore dello stesso bit in <Destinazione>
; funzionamento:
; CF <- bit di <Destinazione> il cui numero è <Op2>
```

```
; <Destinazione> <- <Destinazione>, ma con il bit numero <Op2> = 1
```

Esempi:

```
BTS EAX, 0   ; copia nel carry flag il valore del bit meno significativo
             ; di EAX, poi mette quel bit di EAX a uno
BT Word PTR [variabile], 15   ; CF <- bit più significativo di "variabile"
                              ; "variabile" non viene modificato
..
MOV ESI, 0F80h
BSF EAX, ESI     ; EAX <- 3 (il primo bit a 1 da sinistra
                 ; è quello di peso 3)
```

SHLD, SHRD: Double precision shift (386>)
Istruzioni a tre operandi, utili per effettuare shift di numeri di 64 bit.

```
SHLD <Destinazione>, <RegistroParteBassa>, <Numero di shift>
; Shift Left Double precision
; funzionamento:
; <Destinazione> <- shift a sinistra <Destinazione> <Numero di shift> volte,
; i bit che entrano in <Destinazione> vengono da <RegistroParteBassa>

SHRD <Destinazione>, <RegistroParteAlta>, <Numero di shift>
; Shift Right Double precision
; funzionamento:
; <Destinazione> <- shift a destra <Destinazione> <Numero di shift> volte,
; i bit che entrano in <Destinazione> vengono da <RegistroParteAlta>
```

<Numero di shift> può essere un numero in immediato od il registro CL.

Esempi:

```
SHLD EDX, EAX, 4 ; moltiplica per 16 il numero di 64 bit
                 ; contenuto in EDX, EAX. !!!! NO verificare (in EAX c'è la
                 ; parte alta del risultato ma in EDX NO (provare con NASM)
SHLD EAX, EDX, 4 ; divide per 16 il numero di 64 bit
                 ; contenuto in EDX, EAX. !!!! NO verificare (in EAX c'è la
                 ; parte alta del risultato ma in EDX NO (provare con NASM)
```

SETcc: byte Set on condition (386>)
E' un insieme di 30 istruzioni che, al posto di cc (**c**ondition **c**ode), hanno una condizione. Se cc è vera l'istruzione mette il numero 1 nella destinazione, se è falsa vi mette 0. Le condizioni sono quelle tipiche: Above o Below, Greater o Less, le loro combinazioni con Not e Equal e i flag estesi 386. Questa istruzione è utile per implementare il tipo "boolean" nei linguaggi di programmazione di alto livello.

```
SETcc <Registro o memoria da 8 bit>
; Set operand if "condition code"
; funzionamento:
; <Registro o memoria da 8 bit> <- 1 se cc è vero, 0 se cc è falso
```

Esempi:

```
SETO BYTE PTR [LocOverflow]   ; se l'operazione precedente ha dato overflow (OF = 1)
                    ; scrive un 1 in LocOverflow, altrimenti scrive 0.
```

LFS e LGS (Load pointer using segment register) (386>)
Per i "nuovi" registri di segmento sono stati aggiunte le istruzioni LFS e LGS, analoghe ad LDS, LSS e LES dell'8086 e caricano un puntatore segmentato dalla memoria al registro di segmento indicato.
Sintassi:

```
LGS <Registro>, <Memoria>
; Set operand if "condition code"
; funzionamento:
; <Registro> <- [Memoria]
; GS <- [Memoria + 2] | [Memoria + 4]

LFS <Registro>, <Memoria>
; funzionamento:
; <Registro> <- [Memoria]
; FS <- [Memoria + 2] | [Memoria + 4]
```

Accede alla memoria e carica in <Registro> la parte di offset, che sta all'indirizzo indicato da <Memoria>, e in GS la parte di segmento, che sta a <Memoria> + 2 se l'accesso è a 16 bit, a <Memoria> + 4 se l'accesso è a 32 bit.

Esempi:

```
    LGS EBX, [EBP]   ; Carica in GS i 16 bit che trova all'indirizzo
                     ; SS:(EBP + 4) e in EBX i 32 bit che
                     ; trova all'indirizzo SS:EBP.
    LFS SI, [OffsetIndirizzo] ; carica in SI i 16 bit che stanno a OffsetIndirizzo
                     ; ed in FS i 16 bit a DS:( OffsetIndirizzo + 2)
```

Esistono inoltre 20 istruzioni specifiche, introdotte solo per le operazioni "di sistema" in modo protetto.

80386 SX e DX

Insieme al 386 "completo" (386 DX) l'Intel fornì anche una versione con un bus dei dati di soli 16 bit, un po' come con 8086 e 8088, e la identificò come SX. Questa versione fu realizzata per permettere hardware di minor costo, perché c'era la metà delle linee di dati da implementare nella scheda madre.

80387

Inutile a dirlo, anche il 386 aveva il suo coprocessore, ed anche in questo caso produttori diversi da Intel svilupparono unità compatibili e più potenti.

CPU successive al 386

Dal punto di vista del software, le CPU successive all'80386 non presentano novità sconvolgenti. Le innovazioni vanno in gran parte nell'aumento delle prestazioni, ottimizzando le istruzioni già esistenti, aumentando le dimensioni delle cache interne e facendo in modo che le istruzioni possano essere eseguite parzialmente "in parallelo". Le CPU dal 486 in poi hanno il coprocessore matematico all'interno del chip, per cui le istruzioni del coprocessore non debbono più esser passate all'esterno, attraverso bus che ne rallentano il funzionamento.

MMX e 3Dnow!

Le più significative variazioni rispetto del set d'istruzioni del 386 partono dal 1997, con le istruzioni MMX (**M**ulti**m**edia e**x**tentions, o meglio **M**atrix **M**ath E**x**tentions). MMX aggiunge 57 istruzioni al set d'istruzioni della famiglia 80X86, rendendo disponibili funzionalità che sono state specificamente studiate per l'elaborazione digitale dei segnali.

Le istruzioni del set MMX funzionano con operandi interi ma utilizzano i registri dell'unità interna a virgola mobile. Il "cambio" fra l'esecuzione delle operazioni MMX e quelle a virgola mobile deve essere fatto con un'istruzione esplicita e richiede diversi cicli di memoria. Nell'intento della casa costruttrice, le CPU con MMX dovevano rendere inutili i coprocessori DSP esterni (DSP = **D**igital **S**ignal **P**rocessor) utilizzati nelle applicazioni multimediali (elaborazione delle immagini 3D, riconoscimento del parlato, compressione - decompressione di segnali video).

Variazioni al set d'istruzioni sono state effettuate anche da AMD, che ha introdotto una serie di CPU compatibili con Pentium dotate di MMX "esteso", in grado di funzionare anche in virgola mobile (3Dnow!). Successivamente, a partire dal Pentium III, anche Intel ha introdotto nuove istruzioni a virgola mobile (MMX 2).

486

Il 486 è un 386 potenziato. Il parallelismo interno e quello dei bus sono gli stessi. Nel 486 è presente sul chip il coprocessore aritmetico; inoltre è presente una cache di primo livello e gli accessi ai bus sono stati resi più efficienti. Del 486 sono state introdotte anche versioni a basso consumo, per i computer portatili (486 SL), ed altre che includono un moltiplicatore di frequenza, per cui possono essere "alimentati" con una certa frequenza di clock e funzionare internamente a velocità doppia, tripla o quadrupla (486 DX2, DX4 ..). Questo permette di realizzare schede madre non troppo costose, che lavorano a frequenze di clock "normali", mentre si può spingere fortemente sulla velocità di elaborazione, che viene incrementata dalle frequenze di clock più alte che si possono ottenere all'interno della CPU. Naturalmente ogni accesso alla memoria funziona alla frequenza di clock della scheda madre, cioè esattamente alla stessa velocità per un DX o per un DX4. L'approccio della moltiplicazione del clock interno della CPU, introdotto nel 486, si è poi generalizzato e viene oggi usato praticamente in tutte le CPU nuove, anche nelle famiglie diverse dalla X86.

Pentium e 586

Il Pentium è essenzialmente una CPU a 32 bit. Molto diverso dal 486 nell'architettura interna, dal punto di vista del software è solo una evoluzione, nelle prime versioni non ci sono novità particolari nel set d'istruzioni; nelle successive versioni i Pentium hanno architettura con le nuove istruzioni MMX. Il significato di MMX era stato inizialmente pubblicizzato come "**M**ulti**m**edia e**x**tension", in un secondo tempo fu chiamato "**M**atrix **M**ath e**x**tensions". Dato che i programmi di grafica, e, in generale, multimediali, fanno ampio uso di istruzioni a virgola mobile, l'ultima definizione descrive molto meglio le caratteristiche delle istruzioni MMX, che sono tutte relative a numeri interi.

Nel Pentium la parte "centrale" della CPU ha un funzionamento simile alle CPU di tipo RISC, con enfasi nell'esecuzione delle istruzioni fra registri, un gran numero di registri "nascosti" e grosse cache per le istruzioni e gli indirizzi.

Il set d'istruzioni compatibile con le vecchie CPU della famiglia viene realizzato "sopra" questo "motore" di tipo RISC, quasi come se fosse uno strato di un Sistema Operativo.

L'ALU ha due pipeline da cinque stadi per ogni istruzione su numeri interi, la unità a virgola mobile (FPU: Floating Point Unit) ha una pipeline da otto stadi. La CPU utilizza la previsione dinamica dei salti.

pipeline a 5 stadi (6 in Pentium MMX)

Nei Pentium il registro CR0 contiene i flag PG, CD, NW, AM, WP, NE e ET

Pentium pro e 686
DIB Dual Independent Bus: due bus uno fra processore e memoria principale ed un altro fra processore e cache di secondo livello, permettono trasferimenti contemporanei.
Esecuzione di istruzioni contemporanee (fino a quattro). Esecuzione dinamica = esecuzione non in ordine (out - of - order execution) + esecuzione speculativa (speculative execution)
pipeline a 12 stadi
previsione dei salti (branch prediction) con due tecniche: una basata sulla storia delle istruzioni precedenti (dinamica) ed un'altra sulla conoscenza statistica di come ogni codice operativo è usato più spesso.
cache L1 16 kByte,
Ibrido con cache di livello 2 sulla stessa piastra
Connettore (Socket 8) con 387 piedini Single Edge Contact (S.E.C.)
Funziona a 3,3 V
Il Pentium pro realizza la esecuzione "speculativa" di istruzioni che non è detto che servano.

Caratteristiche avanzate delle nuove CPU **della famiglia 8086**

Pentium II
Da fuori sembra molto diverso, ma di fatto è analogo al Pentium Pro nelle pipeline e nei bus DIB. La differenza più importante, ed anche la più macroscopica, sta nel connettore modulare di nuova concezione ("slot"), non compatibile con le schede madri precedenti.
Nel contenitore ci sono diversi chip e componenti: la CPU vera e propria, la cache di secondo livello (L2 cache) e resistori e condensatori discreti per l'adattamento delle linee.
Connettore (Slot 1) con 242 piedini
frequenze più alte (300 MHz)
MMX: 57 nuove istruzioni che permettono di trattare blocchi di dati da 32, 16 o 8 bit come parole da 64 bit, permettendo l'esecuzione più veloce delle istruzioni tipiche dei programmi multimediali, quali operazioni grafiche e DSP.
Funziona a 2,2 V, perciò si può mandare a frequenze maggiori
cache L1 on chip raddoppiata (32 kByte)
collegamento di due processori Pentium II in multiprocessing senza nessun altro chip di supporto.
La cache può gestire pagine di 4 kByte o di 4 MByte pages. La cache delle istruzioni ha due TLB (Transaction Lookaside Buffer) associativi a quattro vie (four-way, set-associative). Analoghe le caratteristiche della cache dei dati.
Usa tecniche di predizione dinamiche dei salti (dynamic-branch prediction). I salti sono predetti dalla CPU, a differenza della predizione statica, che viene fatta dal compilatore.

Itanium
Intanium non appartiene alla famiglia X86. Ha codici operativi "nativi" diversi da quelli X86, ma

Opteron
A 64 bit di parallelismo interno, è pienamente compatibile con gli X86 a 32 bit, espendendo l'architettura a 64 bit.
Può funzionare nei seguenti modi:
Legacy mode, modalità X86 a 32 bit, si comporta esattamente come un Pentium, ha registri a 32 bit ed il set di istruzioni di un Pentium. Può far girare intoccati i programmi ed i Sistemi Operativi scritti per gli X86 a 32 bit.
Long mode, distingue fra applicazioni a 32 bit ed altre a 64; per le applicazioni a 32 bit si comporta come un Pentium (Compatibility mode), mentre mette a disposizione delle altre la sua architettura estesa da 64 bit (64 bit mode).

Curiosità

Pentium ®
Il 486 non si chiamava 80486, ma 486 e basta! I produttori concorrenti chiamarono 486, con altre sigle a seguire, le loro CPU compatibili con il 486.
Per la generazione successiva Intel provò ad imporre il marchio registrato sulla sigla 586, ma l'ufficio USA competente lo negò, perché non si può registrare come marchio un numero di 3 cifre.
Da allora le CPU Intel non hanno più un numero, ma un vero e proprio "nome", che, naturalmente, è oggetto di trademark.

486 DX e SX
I primi 486 venivano venduti a prezzi molto alti, dato che, integrando il coprocessore aritmetico, davano notevoli incrementi di prestazioni rispetto ai 386. Molti utenti non avevano interesse a passare al 486, perché usavano applicazioni che non richiedevano molti calcoli a virgola mobile (a quei tempi la grafica era un'optional!).
Così Intel produsse una versione del 486, da vendere ad un prezzo molto inferiore, che non permetteva di utilizzare il coprocessore, pur comprendendolo internamente. Questa versione andò sotto il nome di 486 SX.
Quindi un 486 SX è internamente identico ad un DX, in particolare ha lo stesso parallelismo nel data bus, solo che il coprocessore è disabilitato. Naturalmente, per chi si fosse pentito ed avesse voluto in un secondo tempo anche il coproces-

sore per il 486, fu prodotto un 487, che era un normale 486, con dei piedini in più che permettevano di "spegnere" il 486 SX e prendere il suo posto in tutto per tutto.
Queste pratiche di limitazione "artificiale" delle prestazioni di CPU altrimenti identiche continuano tuttora e si potrebbero fare molti altri esempi.

Caratteristiche dei microprocessori della famiglia Intel 80X86

Nome del µp	Produttore	Anno di introduzione	Parallelismo interno (n. ro bit)	Data bus (n. ro bit)	Addr. Bus (n. ro bit)	Memoria virtuale	Frequenza di clock	n. pin	numero di transistor	Novità rispetto al precedente
8088	Intel (IBM, AMD)	1978	16	8	20 (1 MByte)	no	5 MHz fino a 10	DIP	29 000	CPU 16 bit, 1 MByte memoria, predisposta per coprocessore
8086	Intel	Giugno 1978	16	16	20 (1 MByte)	no	5 MHz fino a 10	DIP	29 000	Data bus 16 bit
80186	Intel ed altri	dal 1982 ad oggi	16	16	24 (16 MByte)	no	varie	DIP		Istruzioni più veloci, alcune nuove istruzioni
80286	Intel (IBM, AMD)	Febbraio 1982	16	16	24 (16 MByte)	30 bit (1 GByte)	12 MHz	DIP	134 000	16 MByte memoria, modo protetto segmentato, nuove istruzioni, istruzioni più veloci
80386 SX	Intel	1988	32	16	24 (16 MByte)	46 bit (#) (64 TByte)	16 MHz fino a 33			CPU 32 bit, registri 32 bit, istruzioni 32 bit, modo protetto paginato
80386 DX	Intel	Ottobre 1985	32	32	32 (4 GByte)	46 bit (#) (64 TByte)	16 MHz fino a 33		275 000	32 bit addr.bus (4 GByte memoria)
386 compatibili	AMD, Cyrix ed altri	fino ad oggi	come SX o DX	come SX o DX	come SX o DX	come SX o DX				Oggi sono CPU per sistemi embedded, con I/O. "PC on a chip"
486 SX	Intel	1991	32	32	32 (4 GByte)	46 bit (#) (64 TByte)	25 MHz fino a 33	PGA	1 200 000	microarchitettura nuova, coprocessore interno (disabilitato), istruzioni più veloci
486 DX	Intel	Aprile 1989	32	32	32	46 bit (#) (64 TByte)	33 MHz fino a 50	PGA	1 200 000	Coprocessore interno abilitato, DX2, DX4: moltiplicazione del clock
486 compatibili	AMD, Cyrix, ed altri	fino ad oggi	32	32	32	46 bit (#) (64 TByte)	fino a 100 MHz			CPU per sistemi embedded, con I/O. "PC on a chip"
Pentium (P5)	Intel	Marzo 1993	32	32	32	46 bit (#) (64 TByte)	60 MHz fino a 200	296 PGA	3 100 000	Istruzioni più veloci, cache L1 interna, pipelining, branch prediction
Pentium MMX (P55)	Intel	1997	32	32	32	46 bit (#) (64 TByte)	166 MHz fino a 266	296 PGA	4 500 000	Istruzioni MMX
Pentium Pro (P6)	Intel	1995	32	64	32	46 bit (#) (64 TByte)	150 MHz fino a 200	387 SEC	5 500 000 (*)	microarchitettura nuova, cache L2 su chip separato ma nel contenitore CPU, esecuzione speculativa, slot
Pentium 2	Intel	1997	"	"	"	"	233 MHz fino a 450	242 SEC	7 500 000	Pentium pro + MMX
K6	AMD	1997	"	"	"	"	166 MHz fino a 550	321 PGA	9 300 000	Istruzioni 3Dnow!
Pentium 3	Intel	1999	"	"	"	"	450 MHz fino a 1,3 GHz	370 SEC	9 500 000	Nuove istruzioni MMX, prima slot poi socket

Athlon (K7)	AMD	1999	"	"	"	"	500 MHz fino a 1,2 GHz	SEC	22 000 000	Istruzioni Enhanced 3Dnow!
Pentium 4 (P7)	Intel	2000	"	"	"	"	1,4 GHz	423 PGA	42 000 000	microarchitettura nuova, 20 stadi pipeline, ALU a doppia frequenza, "trace cache" per le istruzioni
Opteron (Hammer)	AMD		64							Nuovi registri e istruzioni a 64 bit, compatibilità con 32 e 16 bit

(*) senza contare i transistor della cache secondaria da 256 kByte, che è su un secondo chip contenuto nello stesso package.
(#) solo con memoria virtuale segmentata (di solito non supportata dal sistema operativo), con memoria "flat": 32 bit di memoria virtuale (4 GByte).

Xeon e Celeron sono nomi diversi di core CPU molto simili ai Pentium 2 e 3, i nomi sono sono stati scelti per ragioni commerciali. Le prestazioni del Celeron sono diminuite artificialmente, quelle degli Xeon spinte al massimo. Agli Xeon sono anche aggiunte funzioni più potenti per il collegamento in sistemi multiprocessore.