


Le architetture INTEL

Alberto Borghese
Università degli Studi di Milano
Dipartimento di Scienze dell'Informazione
email: borghese@dsi.unimi.it

Le prime architetture Intel

1978 – 8086. Estensione del micro-processore 8080 utilizzato per applicazioni industriali. Stessa ISA. Architettura a 16 bit con registri a 16 bit. Parte dei registri è dedicata a compiti specifici, non è un'architettura con registri general-purpose.

1980 – 8087. Coprocessore matematico. Dedicato a velocizzare le operazioni in virgola mobile. Modifica nel modo di gestire gli operandi. Questi potevano essere presi dallo stack oppure, uno di loro, dai registri. Gli operandi venivano presi sempre dalla cima dello stack.
Estensione degli operandi a 10 byte (80bit), *Extended Double Precision*.

Ciclo di esecuzione di un'operazione aritmetica:
Push <operando_1> in stack (esteso a 10byte); Push <operando_2> in stack (esteso a 10byte).
Operazione.
Pop <risultato>.

NB E' il compilatore a potere dichiarare variabili su 10byte (Extended Double).

Limite nello spazio di indirizzamento: 1Mbyte (2^{20}). Questo perché l'indirizzamento veniva operato come: Base shl 4 + Offset.

Gestione di memoria ed I/O tramite 3 segnali di controllo.

Le architetture Intel avanzate

1982 - 80286. L'architettura diventa a 24 bit. Viene utilizzata una modalità di utilizzo protetta che consente di mappare le pagine di memoria in indirizzi privati. Aggiunta di istruzioni specifiche.

1985 – 80386. Estensione a 32 bit. Archtettura, spazio di indirizzamento e registri a 32 bit. Nuove istruzioni, molto vicino ad un calcolatore con general-purpose registers. Pre-fetching. Paginazione della RAM.

1989-1992. 80486, Istruzioni per la gestione delle architetture multi-processore e per il trasferimento di dati condizionato. Cache. Pipe-line singola. Microprogrammazione per l'Unità di controllo (FSM).

1992-1995. Pentium, PentiumPro. Pipe-line multiple (super-scalare). Doppia CPU. Tecnologia MMX (Multi-media extension, SIMD, Pentium). Cache primaria e secondaria (separata con bus dedicato).

1997 – Pentium II. Memorie cache a doppio accesso per ciclo di clock. Cache dei registri di segmento. PentiumPro + MMX.

1999 – Pentium III. "Internet Streaming single instruction multiple data extensions (ISSE). Estensione dell'architettura MMX ad istruzioni floating-point. L2 cache e CPU nello stesso integrato.

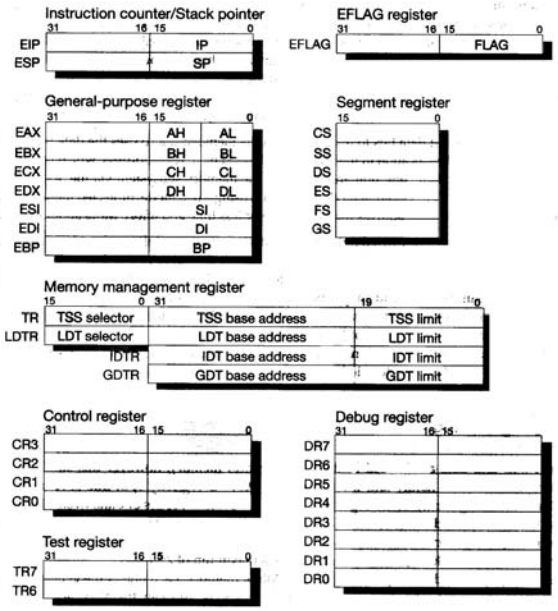
2001 – Pentium 4. Estensione del parallelismo e della Superscalarità. ***Hyper-Threading Technology.***

I registri di un'architettura INTEL

Nascono con 80386.

Sono registri a 32 bit, ma si deve potere accedere ad 1 o 2 byte al loro interno.

I registri di segmento sono rimasti a 16bit.



I registri di segmento

Nome	Descrizione	Utilizzo
CS	Code Segment	Contiene l'indirizzo base dei (dati) ed istruzioni ad accesso immediato. Le istruzioni del segmento sono indirizzate tramite il registro *EIP* (Extended Instruction Pointer). Per modificare *CS* occorre una *chiamata a procedura far* o una *far jump* oppure un interrupt (*int*). In modo protetto viene verificato se il nuovo segmento può essere utilizzato dal task.
DS (EBX)	Data Segment	Contiene i dati del programma. Molte istruzioni quali la *mov* utilizzano questo segmento. E' il segmento in cui sono contenuti i dati di un task.
SS (EBP)	Stack Segment	Quasi del tutto simile allo stack del MIPS. Cresce verso il basso. Contiene i dati locali delle procedure e gli argomenti di chiamata. Contiene anche gli operandi per le operazioni aritmetiche di tipo accumulatore.
ES, FS, GS	Extra Segments	Principalmente utilizzati per operazioni su stringhe. Possono essere utilizzati in sostituzione di DS per accedere a dati al di fuori di DS. Il DOS ed il BIOS utilizzano spesso ES come buffer per le loro chiamate.

Si sovrappongono ai registri a 32 bit (e.g. SS -> [BP, SS] = EBP)

Utilizzo principale dei registri - I

Nome simbolico			Nome descrittivo	Funzioni
32 bit	16 bit	8 bit		
EAX	AX	AH, AL	Accumulator	Moltiplicazione/Divisione, I/O, shift veloce
out 70h, al; Il contenuto di al viene trasferito alla porta 70h.				
EBX	BX	BH, BL	Base Register	Puntatore all'indirizzo base del segmento dati
mov ecx, [ebx]; trasferisci quanto presente all'indirizzo 0($ebx) in ecx (cf. sw $ecx, 0($ebx)).				
ECX	CX	CH, CL	Count Register	Indice di conteggio per cicli, rotazioni e shift
start:	move ecx, 10h out 70h, al loop start		; load ecx con 10h (=16), valore di inizio conteggio (associato a "loop") ; Il contenuto di al viene trasferito alla porta 70h. ; ritorna ad inizio ciclo, il quale verrà ripetuto 16 volte (fino a che ecx = 0)	
EDX	DX	DH, DL	Data Register	Moltiplicazione/Divisione, I/O
mul ebx; moltiplica ebx con eax (implicito), il risultato è contenuto nella coppia edx:eax (hi:lo).				

Utilizzo principale dei registri - stack

Nome simbolico			Nome descrittivo	Funzioni
32 bit	16 bit	8 bit		
ESP	SP	x,x	Stack Pointer	Stack Pointer
EBP	BP,SS	x, x	Base Pointer	Indirizzo base del segmento di Stack (32 bit)

```
        push add1   ; create the first summand (NB automaticamente ESP viene decrementato)
        push add2   ; create the second summand
        push add3   ; create the third summand
addition: proc near  ; call near (inside 64k segment, cf. branch)
        push ebp    ; salva l'indirizzo base per il ritorno (cf. record di attivazione)
        move ebp, esp   ; copia lo Stack pointer nel Base Pointer (individua frame di procedura).
        move eax, [ebp+16]  ; carica sum1 in EAX (NB lo stack anche qui cresce verso il basso).
        add eax, [ebp+12]     ; somma in eax sum1 + sum2
        add eax, [ebp+8]      ; somma in eax aum1 + sum2 + sum3
        pop ebp  ; recupera l'indirizzo base precedente (cf. record di deattivazione)
        ret          ; ritorno al programma chiamante (cf. jr $ra)
addition endp
```


Utilizzo principale dei registri - II

Nome simbolico			Nome descrittivo	Funzioni
32 bit	16 bit	8 bit		
ESI	SI	x, x	Source Index	Indice per la stringa sorgente o indice di caratteri/array
EDI	DI	x,x	Destination Index	Indice per la stringa destinazione o indice di caratteri/array

Esempio: output della stringa: "abcdefghijabcdefghij" su un monitor monocromatico

```
string db 20 dup ('abcdefghijabcdefghij')  ; definizione della stringa
mov eax, @data          ; caricare l'indirizzo (di inizio) dei dati (cioè dell'inizio della stringa) in EAX
mov ds, eax             ; impostare ds a questo segmento dati
mov eax, b800H          ; caricare l'indirizzo del segmento di RAM video in eax
cld                     ; sequenza ascendente
mov ecx, 5              ; trasferisce 5 parole di 4 byte ciascuna
mov esi, OFFSET string  ; carica l'indirizzo della stringa nel segmento esteso (stringa sorgente)
mov edi, 00h            ; carica l'indirizzo del primo carattere in alto a sinistra (stringa destinazione)
movsw                   ; trasferisce 5 parole (20 caratteri)
```

Le operazioni aritmetiche su x386

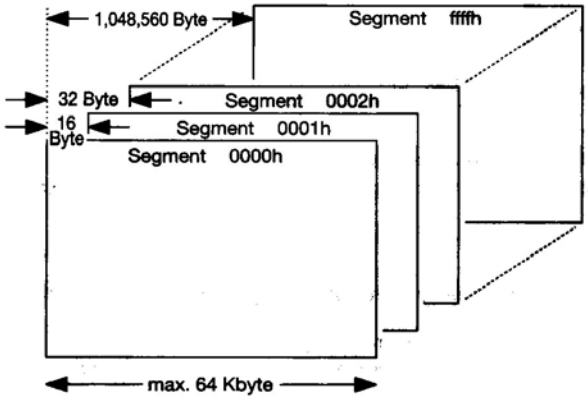
Tipo operando 1	Tipo operando 2	Tipo risultato
Registro	Registro	Registro
Registro	Immediato	Registro
Registro	Memoria	Registro
Memoria	Registro	Memoria
Memoria	Memoria	Memoria

Gli operandi Immediati possono arrivare a 32bit, gli altri ad 80.

Uno dei registri (memoria) deve fungere da operando e destinazione (architettura ad accumulatore). Uno od entrambi gli operandi può provenire direttamente dalla memoria (a differenza che nel MIPS o nel PowerPC).

Indirizzamento in modalità protetta

"Interleaving dei segmenti: spazio di indirizzamento di 1Mbyte suddiviso in segmenti di 64kbyte".



Modalità reale:
Indirizzo = 16 * segmento + Offset (10h*Segment + Offset).

Modalità Virtuale (a partire dal x286):
Indirizzo e offset sono separati.

Modalità di indirizzamento - I

Modo	Descrizione	Restrizione Sui registri	Codice MIPS	Codice INTEL
Indirizzamento diretto (da registro) Register Addressing	Indirizzamento tramite registro (l'operando è in un registro)	No ESP e EBP	move $s0, $s1	move eax, ebx
Base + offset (8 - 32bit)	Base or displacement addressing (INTEL – displacement)	No ESP e EBP	lw $s0, 100($s1)	move eax, array[0]
Base + offset	Base register	No ESP e EBP		move eax, [ebx]
Base + offset	Scaling Factor displacement	No ESP e EBP		move eax, [esi*4]

Esempio: c'è un array di 10 elementi; ciascun elemento ha dimensioni: h x w x d x c. Ciascuno dei 4 elementi è lungo 1 byte. L'array inizia all'indirizzo 0x0c224H. Vogliamo caricare il campo d nell'accumulatore dell'elemento #z.
mov EBX, 0x0c224H
mov ESI, #z
*mov EAX, [EBX + ESI*4 + 2] ; defauld base address is DS.*

Esiste la possibilità di sovrascrivere il segmento dati (*segment override*).
*mov EAX, ES:[EBX + ESI*4 + 2]*

Modalità di indirizzamento - II

Modo	Descrizione	Restrizione Sui registri	Codice MIPS	Codice INTEL
Indirizzamento immediato	L'operando è in una parte dell'istruzione		li $s0, 0x6a02H	move eax,0x6a02H
PC_relative addressing	Indirizzamento relativo al program counter		bne $s0, $s1, label	jnz 0x01A5
Psudodirect addressing	Indirizzo ottenuto cambiando i 26 bit dell'istruzione con i 4 più significativi del PC (i 2 meno significativi sono 00) In INTEL corrisponde a modificare il registro Code Segment.		j label	move cs, 0x87eeae move eip 0x000000

Formato dei dati e degli indirizzi

Dati a 8, 16, 32 byte -> viene scelto un formato preferenziale.
La modifica del formato viene specificata con un **prefisso**.

8086:
Modifica segmento base.
Blocco accesso al bus.
+ move, numero di byte da trasferire.

80386:
Modifica della dimensione degli indirizzi.

- Istruzioni per lo spostamento dei dati (push, pop, utilizzo dello stack e della memori dati).
- Istruzioni aritmetiche logiche, confronto e operazioni.
- Istruzioni di controllo del flusso (basati sui flag, allineamento al byte).
- Le istruzioni della gestione delle stringhe.

Operazioni

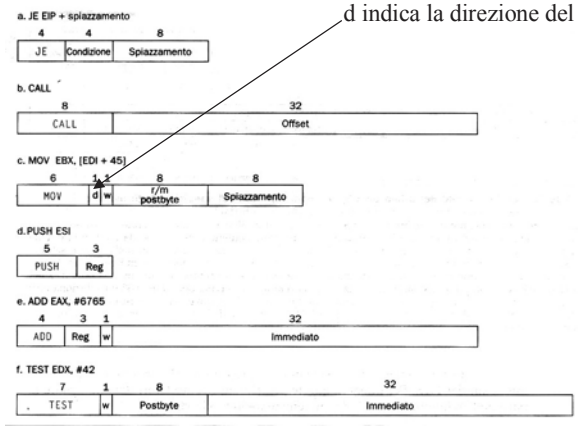
Istruzione	Significato
Controllo	**Salti condizionati e incondizionati**
JNZ, JZ	Salto condizionato a EIP+offset a 8bit; nomi alternativi sono JNE (per JNZ) e JE (per JZ)
JMP	Salto incondizionato; offset a 8 o a 16 bit
CALL	Chiamata a procedura, con offset a 16 bit; l'indirizzo di ritorno è memorizzato nello stack
RET	Prende dallo stack l'indirizzo di ritorno ed esegue il salto
LOOP	Ciclo: decrementa ECX e salta a EIP+offset a 8bit se ECX è diverso da 0
Trasferimento dati	**Spostamento di dati fra registri o fra registri e memoria**
MOV	Sposta un dato da un registro ad un altro o fra registro e memoria
PUSH, POP	Mette nello stack l'operando sorgente; recupera dallo stack un operando e lo mette in un registro
LES	Carica dalla memoria ES ed uno dei GPR
Aritmetiche e logiche	**Operazioni aritmetiche e logiche su dati nei registri o in memoria**
ADD, SUB	Somma il sorgente alla destinazione, sottrae il sorgente alla destinazione; formato registro-memoria
CMP	Confronta il sorgente con la destinazione; formato registro-memoria
SHL, SHR, RCR	Scalamento a sinistra, scalamento a destra, rotazione verso destra con inserimento del flag di carry
CBW	Converte il byte che si trova negli 8 bit meno significativi di EAX in una parola che occupa i 16 bit meno significativi di EAX
TEST	Mette i valori opportuni nei flag facendo l'AND logico fra sorgente e destinazione
INC, DEC	Incrementa la destinazione, decrementa la destinazione; formato registro-memoria
OR, XOR	OR logico, OR esclusivo; formato registro-memoria
Stringhe	**Trasferimenti fra operandi stringhe; lunghezza data da un prefisso di ripetizione**
MOVS	Copia una stringa sorgente in una stringa destinazione incrementando ESI ed EDI; può essere ripetuta
LODS	Carica nel registro EAX un byte, una parola o una double word appartenente ad una stringa

Assembly, esempi

Istruzione	Funzione
JE nome	if equal (condition code) {EIP=nome}; EIP-128 <= nome < EIP+128
JMP nome	EIP=nome;
CALL nome	SP=SP-4; M[SP]=EIP+5; EIP=nome
MOVW EBX, [EDI+45]	EBX=M[EDI+45]
PUSH ESI	SP=SP-4; M[SP]=ESI
POP EDI	EDI=M[SP]; SP=SP+4
ADD EAX, #6765	EAX=EAX+6765
TEST EDX, #42	Setta i flag con il risultato dell'and fra EDX e 42_{esa}
MOVSL	M[EDI]=M[ESI]; EDI=EDI+4; ESI=ESI+4

Codifica delle istruzioni

Molti formati: ampiezza 1-15byte. Codice operativo su 1 o 2 byte.
CLC (Clear Carry è un'istruzione su 1 byte, non ha operandi).
*mov EAX, ind1:[ind2 + ind3*4 + 2] (richiede 17 byte, mod = 2, r/m = 4)*



d indica la direzione del trasferimento

w specifica se lavora sul byte o sulla parola a 32 bit
Post_byte (r/m specifica la modalità di indirizzamento)

Codifica delle istruzioni complesse – registro

reg	w =0	w = 1	
16bit	32bit	16bit	32bit
0	AL	AX	EAX
1	CL	CX	ECX
2	DL	DX	EDX
3	BL	BX	EBX
4	AH	SP	ESP
5	CH	BP	EBP
6	DH	SI	ESI
7	BH	DI	EDI

Campo reg: la sua interpretazione dipende da **w** (= 0 solo registri a 16 bit; =1 dipende se architettura a 16 o 32 bit).

Codifica delle istruzioni complesse – modalità di indirizzamento

reg	w = 0	w = 1		r/m	mod = 0		mod = 1		mod = 2		mod = 3
16b	32b	16b	32b	16b	32b	16b	32b				
0	AL	AX	EAX	0	ind=BX+SI	=EAX	*stesso ind di mod=0*	*stesso ind di mod=0*	*stesso ind di mod=0*	*stesso ind di mod=0*	*come il campo reg*
1	CL	CX	ECX	1	ind=BX+DI	=ECX	"	"	"	"	"
2	DL	DX	EDX	2	ind=BP+SI	=EDX	"	"	"	"	"
3	BL	BX	EBX	3	ind=BP+DI	=EBX	+disp8	+disp8	+disp16	+disp32	"
4	AH	SP	ESP	4	ind=SI	=*(sib)*	SI+disp8	*(sib)*+disp8	SI+ disp8	*(sib)*+disp32	"
5	CH	BP	EBP	5	ind=DI	= disp32	DI+disp8	EBP+disp8	DI+disp16	EBP+disp32	"
6	DH	SI	ESI	6	ind=disp16	=ESI	BP+disp8	ESI+disp8	BP+disp16	ESI+disp32	"
7	BH	DI	EDI	7	ind=BX	=EDI	BX+disp8	EDI+disp8	BX+disp16	EDI+disp32	"

r/m = 0-3. I registri implicati sono riportati nella colonna con mod = 0.
MOD = 3. Indica sempre il registro come nella modalità del campo reg combinato con il bit w.
r/m = 0-2. Indirizzo = Registro Base + Registro spiazzamento.
r/m = 3. Indirizzo = Registro Base + Registro spiazzamento + spiazzamento (nell'istruzione).
Eccezioni:
r/m = 4. MOD = 0,1,2. Seleziona la modalità "scaled index".
r/m = 5. MOD = 1,2. Seleziona EBP + spiazzamento (32 bit).
r/m = 6. MOD = 1,2. Seleziona BP + spiazzamento (16 bit).

Operazioni di I/O

I/O mapped input/output.
•Istruzioni dedicate (in/out).
•Dati presenti nel registro accumulatore (EAX).
•Distinzione tra memoria ed input/output mediante il segnale di controllo M/IO.
•Le periferiche sono viste mediante le porte di I/O.
Spazio di indirizzamento su 16 bit: 64kbyte porte da 1 byte (32k porte da 2 byte; 16k porte da 1 byte).
•Lettura / scrittura avviene verso il device controller (DR), mediante i segnali di controllo di R/W.

Spazio di indirizzamento duplicato: 16 bit (+4 bit di offset) per la RAM e 16 bit per le periferiche nell'8086.

Ciclo di esecuzione di un'istruzione

Fase di fetch: lettura dell'istruzione mediante la coppia: CS:IP dal segmento codice + instruction counter. IP = IP + #byte_istruzione.

Pre-fetch Queue (Coda di pre-fetch). Streaming dal segmento codice di RAM in un buffer fino allo riempimento. L'UC legge il primo byte dell'istruzione dalla coda di pre-fetch e trasferisce un certo numero di byte nell'IR. Questo consente di trasferire da RAM un altrettanto numero di byte.

Decodifica, Esecuzione. Esecuzione è pipe-line e multi-ciclo. Fino a 300 cicli di clock per i task più complessi quali il task switch tramite gate che viene operata in modalità protetta.

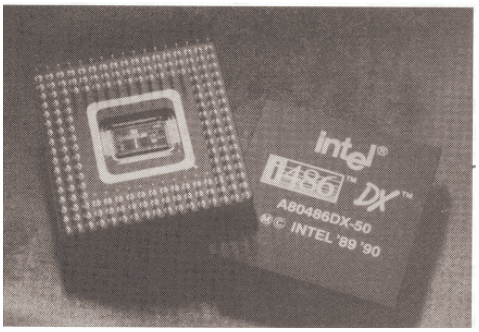
Branch e Jump.

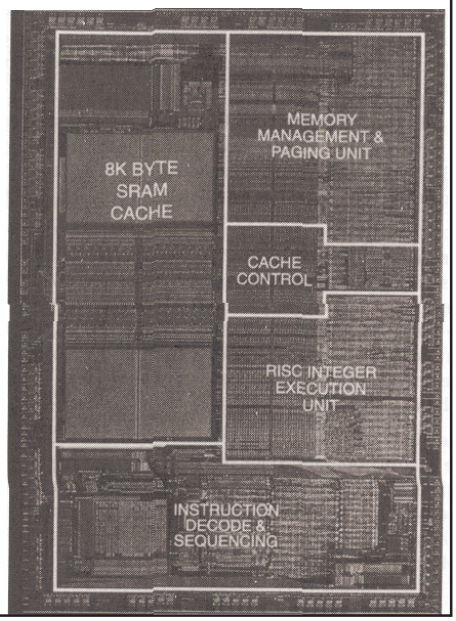
Esempio: Supponiamo che il CS sia 0x80B8, e l'IP 0x019D. L'istruzione successiva sarà all'indirizzo: 0x80B8 019D. A questa istruzione corrisponde il codice 0x7506 che corrisponde a: jnz 0x01A5.

Ci sono due possibilità: 0x01A5 (8 byte dopo), 0x019F.

L'Intel 80486

Integrazione in un solo chip di CPU, Coprocessore e Cache controller.





L'Intel 80486 – Struttura interna

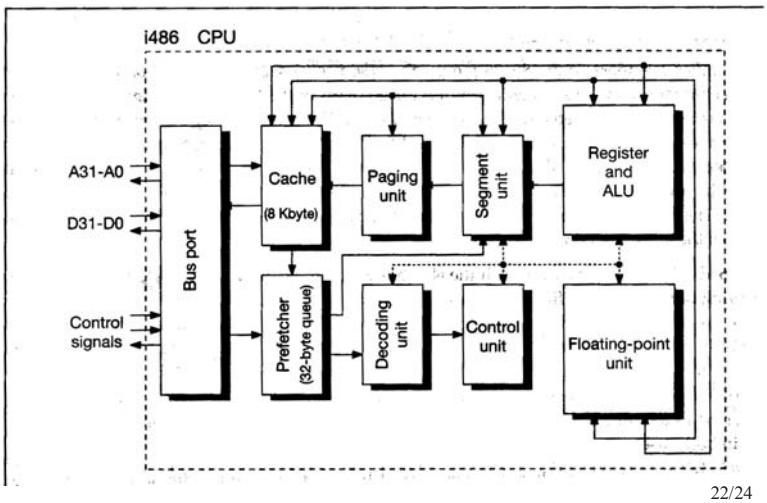
Interfacciamento con il resto della macchina tramite la porta verso il bus. Sul bus vengono inviati i dati, gli indirizzi ed i segnali di controllo.

Pipeline a 5 stadi.

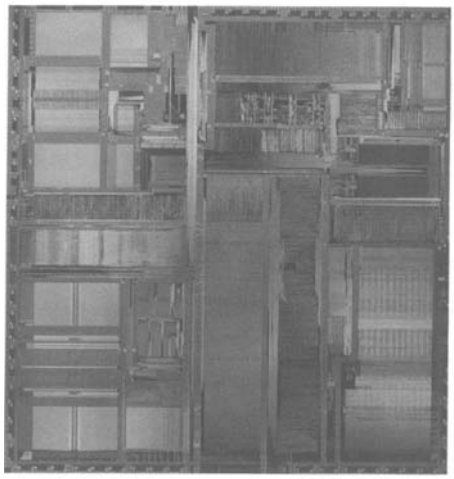
Bus interno a 64bit.

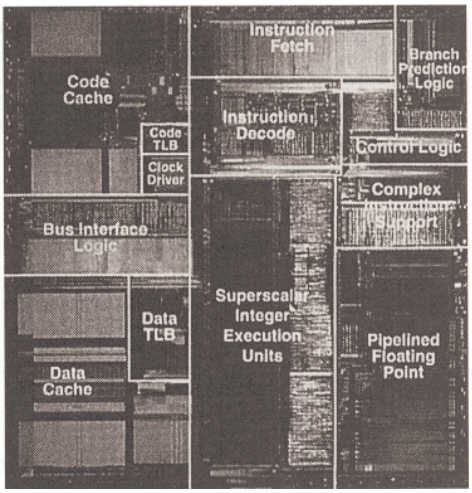
Cache a 8kbyte. Modalità di scrittura: write-through con buffer.

Control unit micro-programmata (cf. macchina a stati finiti).



Il pentium





Pentium – struttura interna

2 pre-fetch queues: la pipeline u si interfaccia con la pipeline floating point.

Advanced Programmable Interrupt Controller (è attivo per la doppia CPU).

Microcodice per l'esecuzione del controllo.

Cache interfacciate sul bus del processore.