Register set supported by Intel Microprocessor along with the historical perspective!

4004, 8008, 8080, 8086 (80186), 80286, 80386, 80486, Pentium 1, 2, Seleron, Pentium-3, 4.

Pentium 4.

64 bit — Dual Core — Hyperthreading

Core, Core 2, Core i 3, 7, 5 Gen-1, 13.

8080 : 8-bit

8086 : 16-bit

80386 : 32-bit.

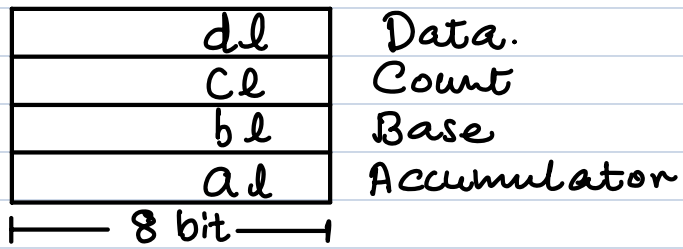
P4 : 64 bit

Core Arch onwards all Intel CPU's are by default 64 bit, but maintain the backward compatibility towards the 32 and the 16 bit.

Stage-1: Registers in 8-bit processors.

there were four registers.

1) Accumulator 2) Base 3) Count 4) Data.



Accumulator Register : expression evaluation.

e.g. High level lang.

$d = a + b - c$

Compute $a + b$.

Store the result $a+b$ in tmp. reg.

Compute : tmp. reg $- c$

Store : tmp. reg $- c$ in some temp reg.

$al \leftarrow a$

$dl \leftarrow b$

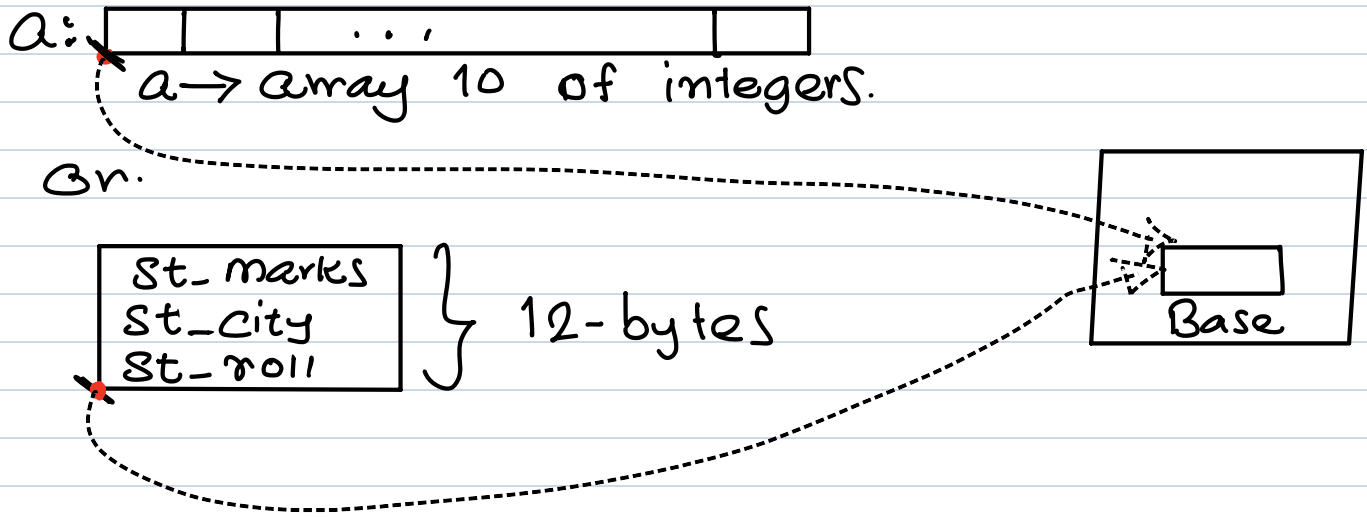
$al \leftarrow al + dl$

$al \leftarrow al - c$

$d \leftarrow al$

---

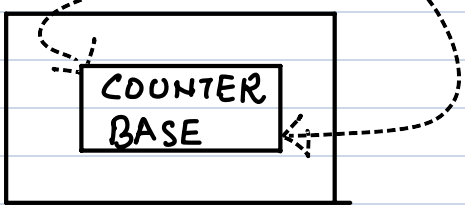
2) Base Register : Role: to store the base address of aggregate data blocks.



3) Count Register :

If you want to maintain counter variable of loop into a register then think of the counter register.

```
for (i=0; i<N; ++i)
{
    a[i] = RHS;
}
```



4] Data Register:

If you want to maintain a data element in register rather than in memory for efficiency then think of the data register

Note : a2, b2, c2, d2 : General Purpose Registers.

8080 ⟶ 8086
8-bit 16-bit

80386 80386
LLP

80386 80387

8 bit	8 bit	
dh	dl	: dx
ch	cl	: cx
bh	bl	: bx
ah	al	: ax
		: si (source index reg)
		: di (dest. index reg)
		: bp (frame pointer reg)
		: sp (stack pointer reg).
		: ip (instruction pointer).

1) 8 bit ⟶ 16 bit

2) Procedure : Assembly level support.

3) String manipulation: Assembly level support

Instructions
+
Registers.

8086 → 80386.

8 bit	8 bit	8 bit	8 bit	
		dh	dl (dx)	: edx
		ch	cl (cx)	: ecx
		bh	bl (bx)	: ebx
		ah	al (ax)	: eax
		si		: esi
		di		: edi
		bp		: ebp
		sp		: esp.
		ip		: eip.

reg8 : 8 : al, bl, cl, dl, ah, bh, ch, dh

reg16 : 8 : ax, bx, cx, dx, si, di, sp, bp

reg32 : 8 : eax, ebx, ecx, edx, esi, edi, esp, ebp.

Corporate training 2500/-

2500/- 2022 2500/- 6,7L

3500/-

CH 4k

8086 16 bit : 20 bits

80386 32 bit : 64 bit.

Category:

G.P.R.

reg8	reg16	reg32
al	ax	eax
bl	bx	ebx
cl	cx	ecx
dl	dx	edx
ah	si	esi
bh	di	edi
ch	sp	esp
dl	bp	ebp

flag register (PSW)

flags : 8086

eflags : 80386

FPU register

#nr = 8

R7 st(0)
R6 st(1)
R5 st(2)
R4 st(3)
R3 st(4)
R2 st(5)
R1 st(6)
R0 st(7)

80 bit

IEEE
- single precision (4 byte)
- double precision (8 byte)
- extended precision. (10 bytes) 80 bit.

mmX register (P2)

multi-media extension 64 bit.

mm0 mm7. : 8

B7	B6	B5	B4	B3	B2	B1	B0

64 bit

SSE : Streaming SIMD extension.

SIMD = Single Instruction Multiple Data.

128 bit. = 16 byte

15														
B15	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0

Xmm0 Xmm7 : 8 - 128 bit

reg

1 byte 2 4

B7	B0

8 - 1 byte Integer

4 - 2 byte Integer.

2 - 4 byte integer.

1 - 8 byte integer

2 - floats.

1 - double.

128 - bit.

16 — 1 byte int.	4 - floats
8 — 2 byte int.	2 - doubles
4 — 4 byte int.	1 — extended double precision
2 — 8 byte	
1 — 16 byte	

```
int a[100];
int b[100];
int c[100];
int i;
for (i = 0; i < 100; ++i)
    c[i] = a[i] + b[i];        // 100.
```

$$\left.\begin{array}{l} r0 \leftarrow a[i] \\ r1 \leftarrow b[i] \\ r0 \leftarrow r0 + r1 \\ c[i] \leftarrow r0 \end{array}\right\} \quad 0 \leq i < 100$$

Xmm0	a[3]	a[2]	a[1]	a[0]
	SPECIAL SSE Add			
Xmm1	b[3]	b[2]	b[1]	b[0]
Xmm2	a[3]+b[3]	a[2]+b[2]	a[1]+b[1]	a[0]+b[0]
	↓	↓	↓	↓
	c[3]	c[2]	c[1]	c[0]

SIMD

SSE

SSE 4.2

4.3