

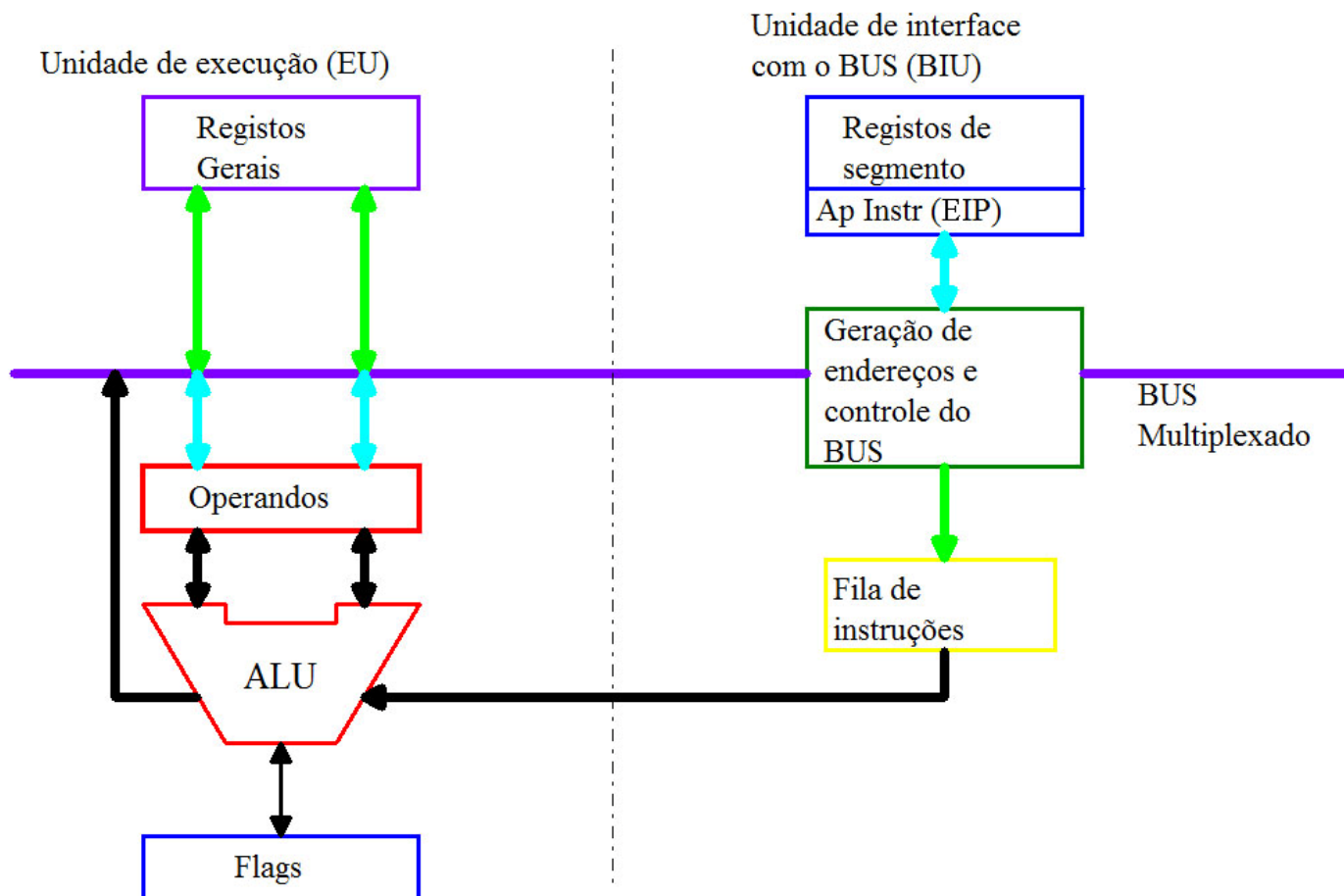
x86

arquitetura e instruções básicas

Família x86

Data	CPU	Palavra	Endereço (bits)
1978	8086 , 8088	16	20
1982	80186 , 80188		
	80286	16	24
1985	80386	32	32
1989	80486		
1993	Pentium , Pentium MMX		
1995	Pentium Pro	32	36
1997	Pentium II/III , Celeron , Xeon		
2003	Pentium M , Intel Core (2006)	32	36
2000	Pentium 4		
2005	Pentium 4 Prescott , Celeron D , Pentium D	64	36
2006	Core 2	64	36
2008	Core i3 , Core i5 and Core i7 (Nehalem / Westmere)	64	36
2011	Intel Core i3 , Core i5 and Core i7 (Sandy Bridge / Ivy Bridge)	64	40
2013	Intel Core i3 , Core i5 and Core i7 (Haswell / Broadwell)	64	44
2015/2016	Intel Core i3 , Core i5 and Core i7 (Skylake / Kaby Lake / Cannonlake)		

Arquitetura

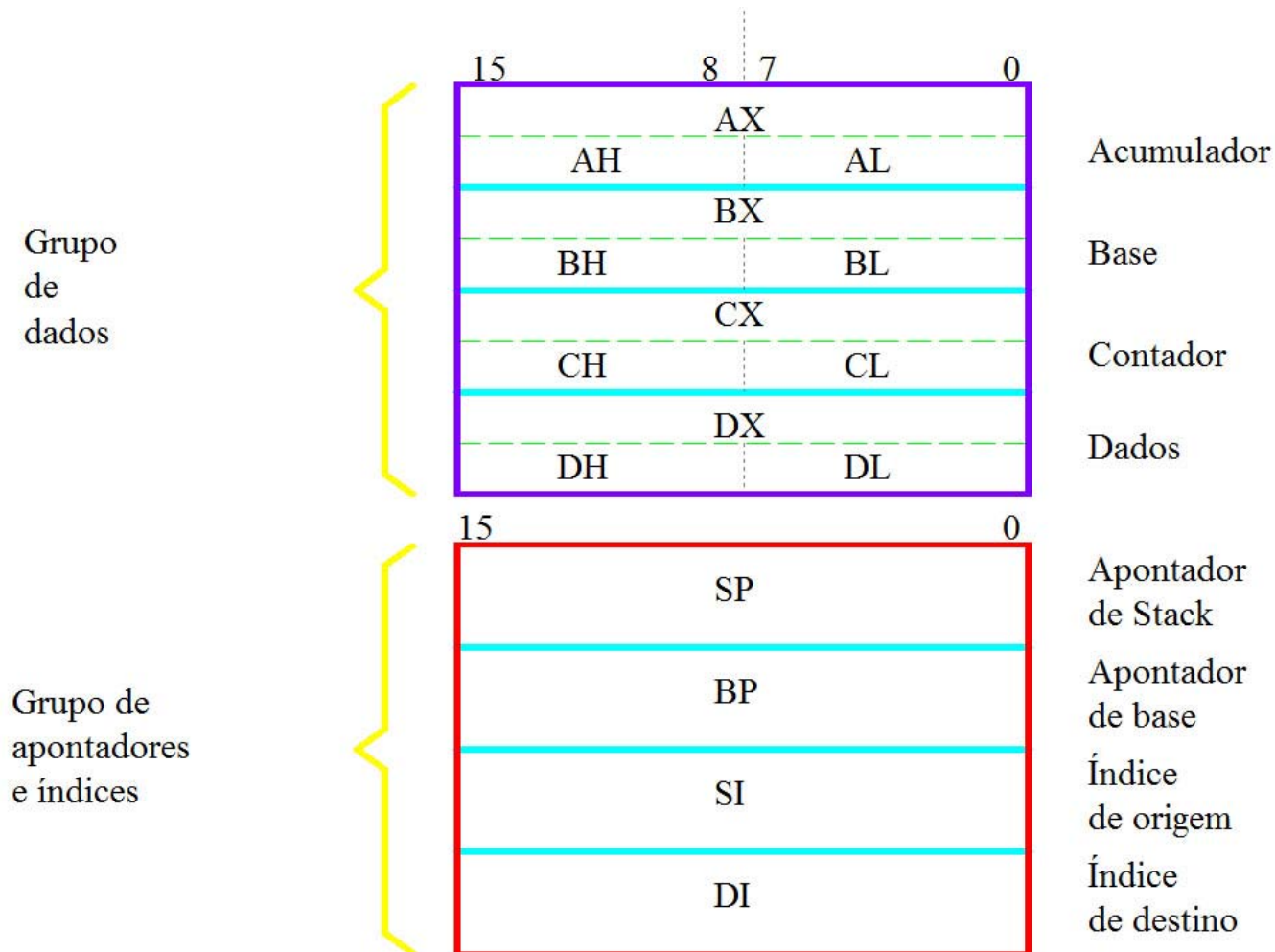


Arquitetura

- A BIU executa todas as operações com o BUS do sistema:
 - Troca de dados entre a EU e a memória do sistema e dispositivos de I/O.
 - Enquanto a EU está ocupada a executar instruções, a BIU lê na memória central as próximas instruções a realizar (prefetch).

Registos gerais (16 bits)

até ao 80286



Registros gerais (32 bits)

31	8	15	8	7	0
Alternate name	AX				
	AH		AL		
EAX					
Alternate name	BX				
	BH		BL		
EBX					
Alternate name	CX				
	CH		CL		
ECX					
Alternate name	DX				
	DH		DL		
EDX					
Alternate name	BP				
EBP					
Alternate name	SI				
ESI					
Alternate name	DI				
EDI					
Alternate name	SP				
ESP					

Registos

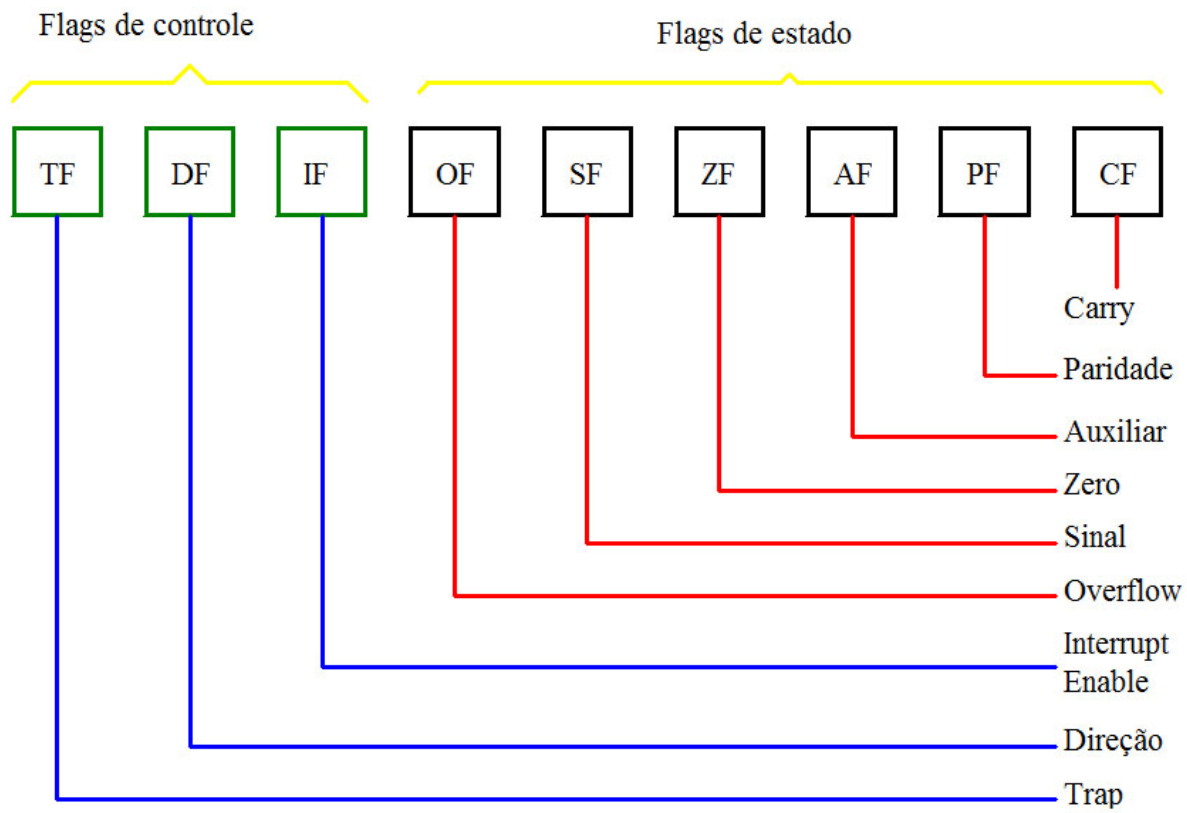
Algumas instruções usam os registos de forma implícita

Registo	Operações
EAX	Multiplicação, divisão, I/O (DoubleWord)
AX	Multiplicação, divisão, I/O (Word)
AL	Multiplicação, divisão, I/O (Byte)
AH	Multiplicação, divisão (Byte)
EBX	Tradução
ECX	Loops, operações com strings
CL	Rotação e deslocamento de variáveis
EDX	Multiplicação, divisão, (Word); I/O indirecto
ESP	Operações de stack
ESI	Operações com strings
EDI	Operações com strings

Flags

O 8086 tem 3 flags de controle que podem ser alteradas pelos programas para alterar as operações do processador, e 6 flags de estado que refletem certas propriedades do resultados das operações aritméticas ou lógicas.

Flags



Flags

- carry - carry ou borrow no bit mais significativo (MSB) de um resultado (8 ou 16 bits)
- paridade - paridade par quando activa
- auxiliar - carry ou borrow entre *nibbles* (usada por aritmética decimal)
- zero - o resultado da operação é 0
- sinal - o MSB é 1 (número negativo)
- overflow - perda de um bit significativo
- interrupt enable - permite que o CPU reconheça interrupções externas (mascaráveis)
- direcção - quando activa provoca o auto-decremento nas intruções sobre strings, de contrário provoca o auto-incremento
- trap - põe o processador no modo *single-step* para *debugging*

Stack (32 bits)

- A **stack** no x86 encontra-se em memória e é referenciada pelo registo ESP (stack pointer).
- Um sistema pode ter um número ilimitado de **stacks**. Apenas se pode endereçar uma **stack** de cada vez.
- As operações na **stack** são sempre sobre 32 bits.
- Um valor é carregado (PUSHed) em **stack** decrementando o ESP 4 unidades e escrevendo o valor na **stack**.
- Um valor é retirado (POPed) da **stack** lendo-o do topo da **stack** e incrementando o ESP 4 unidades.

Instruções de transferência de dados

Âmbito Geral	
MOV destino, origem	Move byte, word, doubleword ou quadword
PUSH origem	Carrega doubleword na stack
POP destino	Retira doubleword da stack
XCHG destino, origem	Troca byte, word, doubleword ou quadword
XLAT tabela	Traduz byte
Transferência de flags	
LAHF	Coloca as flags em AH
SAHF	Coloca AH nas flags
PUSHF	Carrega as flags na stack
POPF	Retira flags da stack

Instruções aritméticas

Adição	
ADD destino, origem	Adiciona byte, word, doubleword ou quadword
ADC destino, origem	Idem com carry
INC destino	Incrementa byte, word, doubleword ou quadword 1 unidade
Subtração	
SUB destino, origem	Subtrai byte, word, doubleword ou quadword
SBB destino, origem	Idem com borrow
DEC destino	Decrementa byte, word, doubleword ou quadword 1 unidade
NEG destino	Nega byte, word, doubleword ou quadword
CMP destino, origem	Compara byte, word, doubleword ou quadword
Multiplicação	
MUL origem	Multiplica byte, word, doubleword ou quadword sem sinal
IMUL origem	Multiplicação inteira de byte, word, doubleword ou quadword
Divisão	
DIV origem	Divide byte, word, doubleword ou quadword sem sinal
IDIV origem	Divisão inteira de byte, word, doubleword ou quadword
CBW	Converte byte em word
CWD	Converte word em doubleword

Instruções de manipulação de bits

Lógicas	
NOT destino	Complementa byte, word, doubleword ou quadword
AND destino, origem	Produto lógico de byte, word, doubleword ou quadword
OR destino, origem	Soma lógica de byte, word, doubleword ou quadword
XOR destino, origem	Ou exclusivo de byte, word, doubleword ou quadword
TEST destino, origem	Teste (AND) de byte, word, doubleword ou quadword
Deslocamentos	
SHL / SAL destino, count	Deslocamento lógico / aritmético de byte, word, doubleword ou quadword à esquerda
SHR destino, count	Deslocamento lógico de byte, word, doubleword ou quadword à direita
SAR destino, count	Deslocamento aritmético de byte, word, doubleword ou quadword à direita
Rotações	
ROL destino, count	Roda byte, word, doubleword ou quadword à esquerda
ROR destino, count	Roda byte, word, doubleword ou quadword à direita
RCL destino, count	Roda byte, word, doubleword ou quadword com carry à esquerda
RCR destino, count	Roda byte, word, doubleword ou quadword com carry à direita

Instruções sobre strings

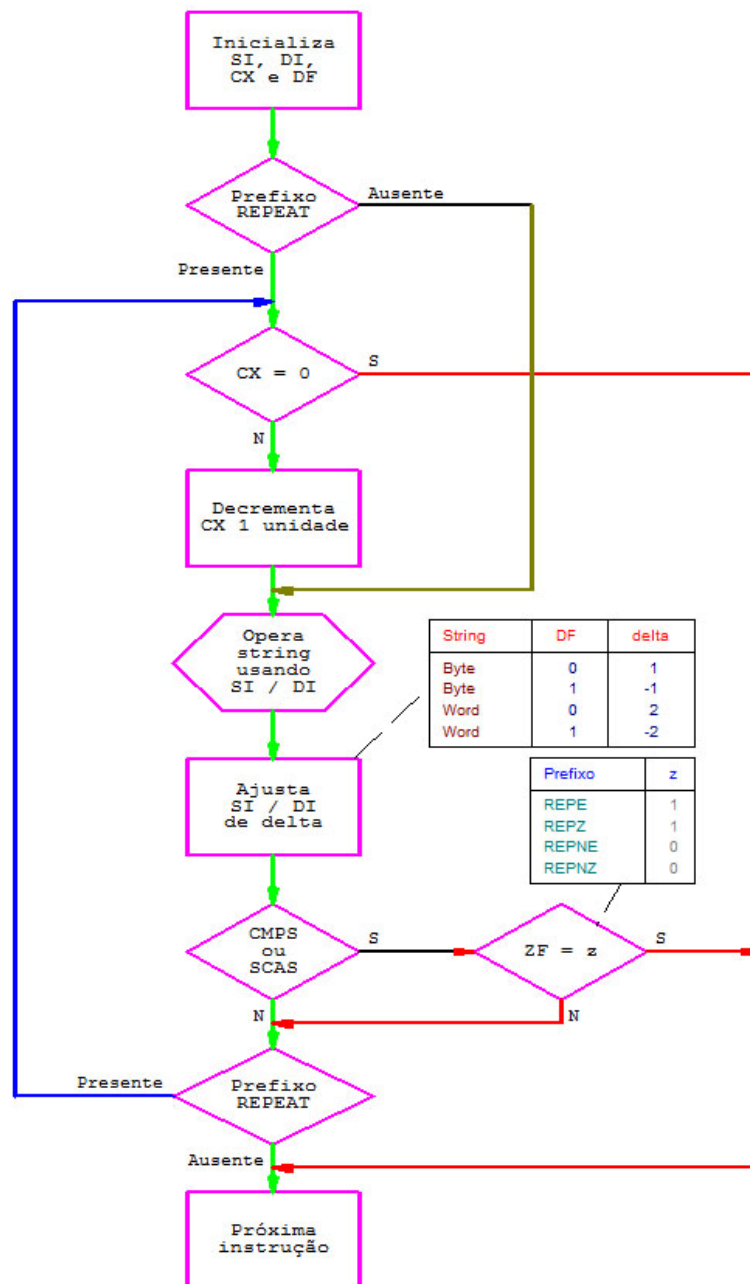
REP	Repete
REPE / REPZ	Repete enquanto for igual / zero
REPNE / REPNZ	Repete enquanto não for igual / não zero
MOVS destino, origem	Move string de bytes, words ou doublewords
MOVSB / MOVSW	Move string de bytes, words ou doublewords
CMPS destino, origem	Compara string de bytes, words ou doublewords
SCAS destino	Pesquisa string de bytes, words ou doublewords
LODS origem	Lê string de bytes, words ou doublewords
STOS destino	Escreve string de bytes, words ou doublewords

Instruções sobre strings

Registos e flags utilizados

ESI	Índice (offset) da string origem
EDI	Índice (offset) da string destino
ECX	Contador de repetições
AL / AX / EAX	Valor para pesquisa Destino para LODS Origem para STOS
DF	0 = auto-incrementa SI, DI 1 = auto-decrementa SI, DI
ZF	Termina pesquisa ou comparação

Fluxo das operações com strings (16 bits)



Instruções de transferência de controle (1)

Transferências condicionais	
JA / JNBE rótulo	Salta se superior / não inferior nem igual
JAE / JNB rótulo	Salta se superior ou igual / não inferior
JB / JNAE rótulo	Salta se inferior / não superior nem igual
JBE / JNA rótulo	Salta se inferior ou igual / não superior
JC rótulo	Salta se carry
JE / JZ rótulo	Salta se igual / zero
JG / JNLE rótulo	Salta se superior / não inferior nem igual
JGE / JNL rótulo	Salta se superior ou igual / não inferior
JL / JNGE rótulo	Salta se inferior / não superior nem igual
JLE / JNG rótulo	Salta se inferior ou igual / não superior
JNC rótulo	Salta se não carry
JNE / JNZ rótulo	Salta se não for igual / não zero
JNO rótulo	Salta se não overflow
JNP / JPO rótulo	Salta se não paridade / paridade ímpar
JNS rótulo	Salta se não tiver sinal
JO rótulo	Salta se overflow
JP / JPE rótulo	Salta se paridade / paridade par
JS rótulo	Salta se tiver sinal

Instruções de transferência de controle (2)

Transferências incondicionais	
CALL nome-de-rotina	Chama procedimento
RET valor-a-retirar (op)	Volta de um procedimento
JMP endereço	Salta
Controle de iteração	
LOOP rótulo	Ciclo
LOOPE / LOOPZ rótulo	Ciclo se igual / zero
LOOPNE / LOOPNZ rótulo	Ciclo se não igual / não zero
JCXZ rótulo	Salta se CX = 0
Interrupções	
INT tipo	Interrupção
INTO	Interrupção se overflow
IRET	Volta de interrupção

Instruções de transferência de controle

(condições testadas)

Mnemónica	Condições testadas
JA / JNBE	$(CF \text{ ou } ZF) = 0$
JAE / JNB	$CF = 0$
JB / JNAE	$CF = 1$
JBE / JNA	$(CF \text{ ou } ZF) = 1$
JC	$CF = 1$
JE / JZ	$ZF = 1$
JG / JNLE	$((SF \text{ xor } OF) \text{ ou } ZF) = 0$
JGE / JNL	$(SF \text{ xor } OF) = 0$
JL / JNGE	$(SF \text{ xor } OF) = 1$
JLE / JNG	$((SF \text{ xor } OF) \text{ ou } ZF) = 1$
JNC	$CF = 0$
JNE / JNZ	$ZF = 0$
JNO	$OF = 0$
JNP / JPO	$PF = 0$
JNS	$SF = 0$
JO	$OF = 1$
JP / JPE	$PF = 1$
JS	$SF = 1$

Instruções de controlo do processador

Operações com flags	
STC	Ativa flag de carry
CLC	Desativa flag de carry
CMC	Complementa flag de carry
STD	Ativa flag de direcção
CLD	Desativa flag de direcção
STI	Ativa flag de interrupt enable
CLI	Desativa flag de interrupt enable
Sincronização externa	
HLT	Para até interrupção ou reset
WAIT	Espera até pino TEST/ estar ativo
ESC	Escape para processador externo
LOCK	Prende o bus para a próxima instrução
Sem efeito	
NOP	Não efectua nenhuma operação

Endereçamento

Operandos em registo e imediato

As instruções com operandos apenas em registo são as mais compactas e rápidas. São executadas inteiramente na CPU.

Exemplo: `ADD AX, BX`

Os operandos imediatos são constantes contidas na instrução, sendo por isso de acesso rápido. Servem apenas como origem, nunca como destino.

Exemplo: `ADD AL, 5`

Endereçamento directo

É o modo de endereçamento mais simples.
É usado para endereçar variáveis simples (escalares).

Exemplos: ADD CX, alfa
 ADD alfa, 6

Endereçamento indirecto

Uma instrução pode operar em diferentes posições de memória se o registo de base ou índice for actualizado.

Exemplos: `ADD BL, [BX]`
 `ADD [SI], 12`

Tipos de dados

C ou Java (depende do exemplo)	Assembly
byte n;	n resb 1
short p;	p resw 1
int x;	x resd 1
long r;	r resq 1
byte n=3;	n db 3
short p=5;	p dw 5
int x=0x31;	x dd 0x31
long r=0x1A;	r dq 0x1A
char s[20];	s resb 20
int v[12];	v resd 12
String t="Olá Mundo";	t db "Olá Mundo"