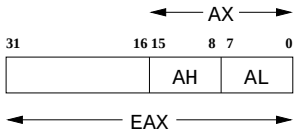
x86 Assembly Reference Sheet

32-bit	16-bit	8-bit high	8-bit low
EAX	AX	AH	AL
EBX	BX	BH	BL
ECX	CX	CH	CL
EDX	DX	DH	DL
ESI	SI		
EDI	DI		
EBP	BP		
ESP	SP		



`jb`	below	CF is set
`jbe`	below or equal	CF or ZF is set
`je`	equal	ZF is set
`jl`	less	SF ≠ OF
`jle`	less or equal	(SF ≠ OF) or ZF is set
`jo`	overflow	OF is set
`jp`	parity	PF is set (even parity)
`js`	sign	SF is set (negative)

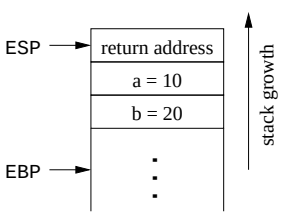
```
movb  (%ebp),%al            # AL ← M[EBP]
movb  -4(%esp),%al          # AL ← M[ESP - 4]
movb  (%ebx,%edx),%al       # AL ← M[EBX + EDX]
movb  13(%ecx,%ebp),%al     # AL ← M[ECX + EBP + 13]
movb  (,%ecx,4),%al         # AL ← M[ECX * 4]
movb  -6(,%edx,2),%al       # AL ← M[EDX * 2 - 6]
movb  (%esi,%eax,2),%al     # AL ← M[ESI + EAX * 2]
movb  24(%eax,%esi,8),%al   # AL ← M[EAX + ESI * 8 + 24]
movb  100,%al               # AL ← M[100]
movb  label,%al             # AL ← M[label]
movb  label+10,%al          # AL ← M[label+10]
movb  10(label),%al         # NOT LEGAL!

movb  label(%eax),%al       # AL ← M[EAX + label]
movb  7*6+label(%edx),%al   # AL ← M[EDX + label + 42]

movw  $label,%eax           # EAX ← label
movw  $label+10,%eax        # EAX ← label+10
movw  $label(%eax),%eax     # NOT LEGAL!

call  printf                # (push EIP), EIP ← printf
call  *%eax                 # (push EIP), EIP ← EAX
call  *(%eax)               # (push EIP), EIP ← M[EAX]
call  *fptr                 # (push EIP), EIP ← M[fptr]
call  *10(%eax,%edx,2)      # (push EIP), EIP ←
                            #         M[EAX + EDX*2 + 10]
```

Conditional branch sense is inverted by inserting an "N" after initial "J," *e.g.*, JNB. Preferred forms in table below are those used by debugger in disassembly. Table use: after a comparison such as

```
cmp %ebx,%esi   # set flags based on (ESI - EBX)
```

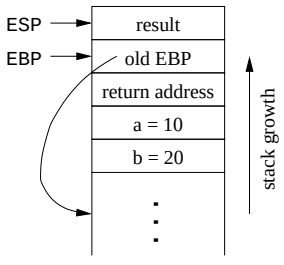
choose the operator to place between ESI and EBX, based on the data type. For example, if ESI and EBX hold unsigned values, and the branch should be taken if ESI ≤ EBX, use either JBE or JNA. For branches other than JE/JNE based on instructions other than CMP, check the branch conditions above instead.

	`jnz`	`jnae`	`jna`	`jz`	`jnb`	`jnbe`	unsigned comparisons
preferred form	`jne`	`jb`	`jbe`	`je`	`jae`	`ja`	
	≠	<	≤	=	≥	>	
preferred form	`jne`	`jl`	`jle`	`je`	`jge`	`jg`	signed comparisons
	`jnz`	`jnge`	`jng`	`jz`	`jnl`	`jnle`	



stack at start of function

```
/* the function */
/* (returns int in EAX) */
int a_func (int a, int b);

/* local variable */
int result;

/* the call */
a_func (10, 20);
```



stack during function execution