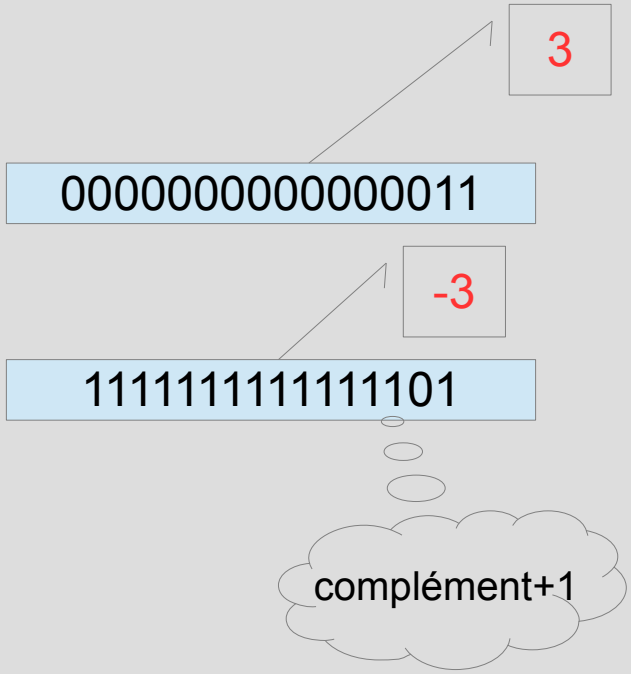
Éléments de langage

Les types de données

Exemple sur (16bits)

Entiers Positifs : 1, 2, …, 32767

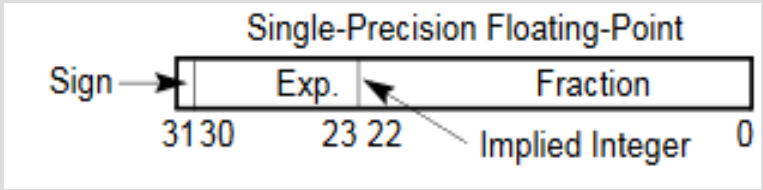
Entier Negatifs : -1, -2, …, -32768

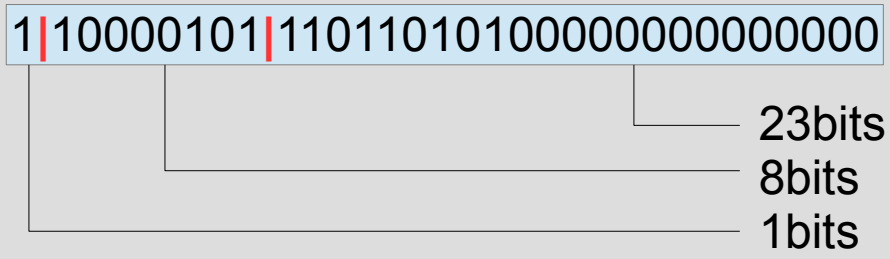


Nombre en virgule flotante (32bits)
Dans une FPU et non en ALU

http://home.agh.edu.pl/~amrozek/x87.pdf

−118,625





Algo :
https://www.youtube.com/watch?v=8afbTaA-gOQ
https://fr.wikipedia.org/wiki/IEEE_754

Chaine d'octets

texte1 db 'ceci est un texte'

Instructions spécialisées pour la manipulation de chaînes (aussi les tableaux)

Parcours du tableau

. Sens de parcours de la chaîne/du tableau :

. défini par le flag DF (Direction Flag)

. Instructions pour spécifier le sens :

. cld (Clear Direction) : incrémentation si/di

. std (Set Direction) : décrémentation si/di

Lire en mémoire

- lodsb
- Remplit le registre al avec l'octet pointé par si
- Incrémente/décrémente si
- lodsw : remplit ax , si±2
- b=byte (1 octet), w=word (2 octets)

Ecrire en mémoire

- stosb
- Le contenu de al est copié dans la case pointée par di
- Incrémente/décrémente di
- stosw : contenu de ax et di±2

Transfert de mémoire à mémoire

. movsb

. l'octet de la zone mémoire pointée par si est copié dans la zone mémoire pointée par di

. Incrémente/décrémente si et di

. movsw : transfert de 2 octets, si et di ± 2

Itération

Le préfixe d'instruction rep :

. Pour répéter une instruction de chaîne un certain nombre de fois

. Le nombre d'itérations est fixé par cx

Exemple:

```
mov cx, 5
rep movsb
```

(répète 5 fois l'instruction movsb)

Longueur d'une chaîne

Pour récupérer la longueur d'une chaîne :

. au moment de la déclaration :

```
ch1 db 'un texte'
long equ $ - ch1
```

Comparaison de chaînes

Comparaison de chaînes de caractères :

. cmpsb

. compare l'octet en si avec l'octet en di

. affecte le flag ZF

. ZF =1 si [si] =[di]

. ZF =0 sinon

. Incrémente/décrémente si et di

. cmpsw : compare 2 octets, si et di ± 2

Recherche d'un caractère dans une chaîne

. scasb

. compare le contenu du registre al avec l'octet en di

. affecte le flag ZF

. ZF =1 si al =[di]

. ZF =0 sinon

. Incrémente/décrémente di

. scasw : recherche ax, di±2

Variantes du préfixe rep:

. repe

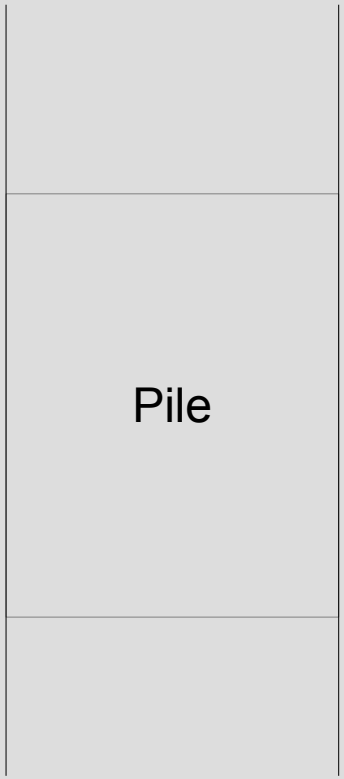
Répète l'instruction qui suit tant que ZF est allumé, ou au maximum cx fois

. repne

Répète l'instruction qui suit tant que ZF est éteint, ou au maximum cx fois

La pile

- La pile est une zone mémoire qui sert à stocker des informations.
- Fonctionne en mode LIFO (Last In First Out)



Les pointeurs de pile

- SP = adresse du sommet de la pile
- BP = adresse du fond de la pile
- SS = adresse du segment de pile

Empiler, Depiler

- push arg : empiler arg (reg., case mémoire, valeur)
- pop arg : dépiler dans arg (reg., case mémoire)

- c'est l'argument qui détermine la taille

Les procedures

name PROC

Instructions

ENDP

Exemple :

Factoriel PROC

.....

ENDP

Appel de procedure

- CALL *nom*

Ex : CALL factoriel

Passage de parametres

Comment passer des paramètres ?

- grâce aux registres ! C'est possible mais très limité
- grâce à la pile ! pas de limitation (sauf la taille du segment)

(1) on empile les paramètres

(2) on appelle la procédure

Passage de parametres

Exemple :

Registre

- Mov ax,10
- CALL factoriel

Pile

- Push 10
- CALL factoriel

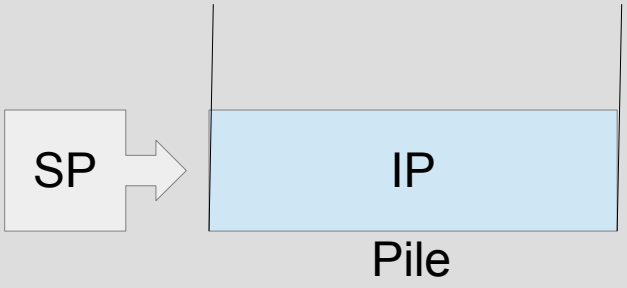
Description

Cette instruction force le microprocesseur à exécuter les instructions du sous-programme indiqué par l'adresse d'appel avant de continuer. Dès que la routine est terminé, l'exécution reprendra son cours à l'instruction suivant le CALL.

Appel court :

PUSH IP

IP ← Destination

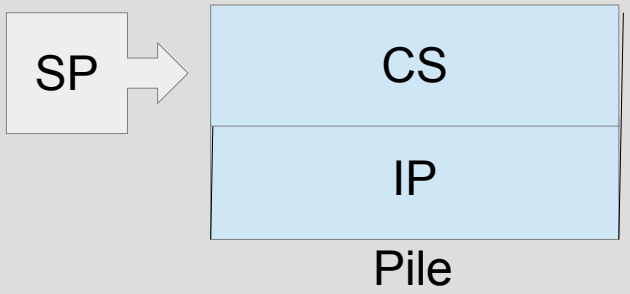


Appel long :

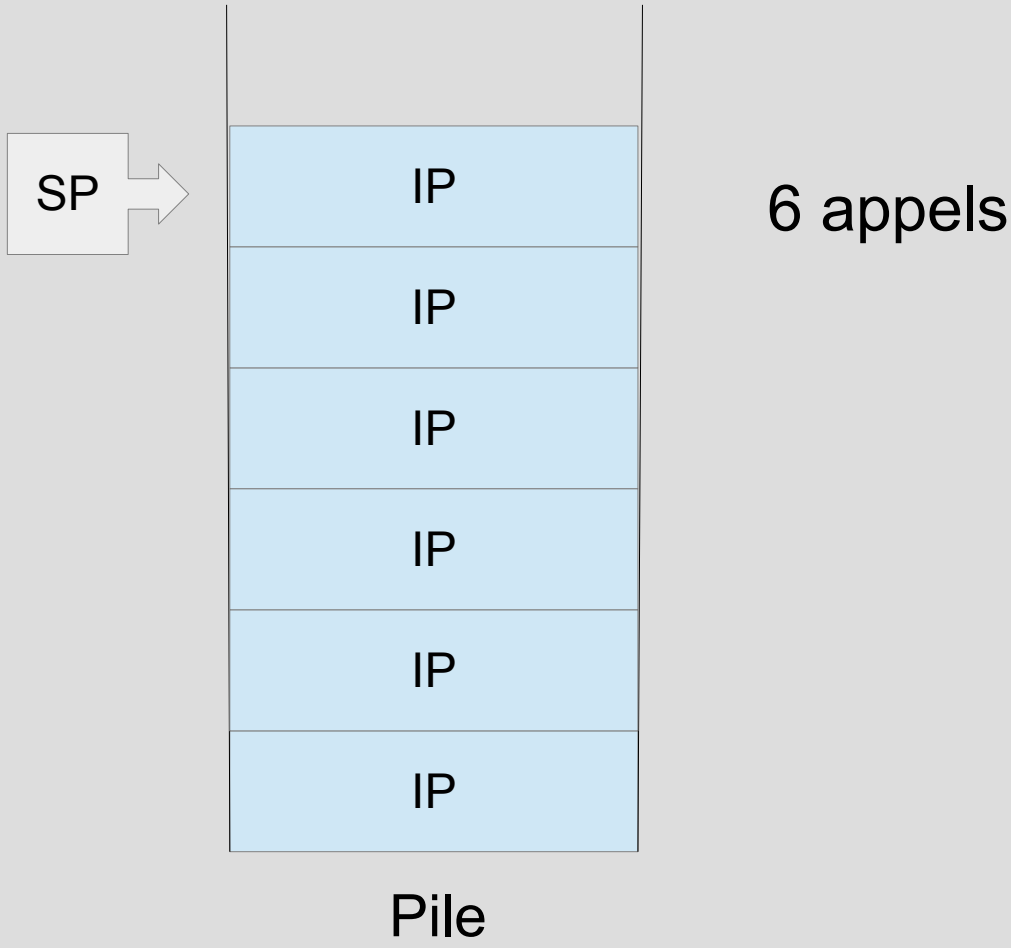
PUSH CS

PUSH IP

CS:IP ← Destination

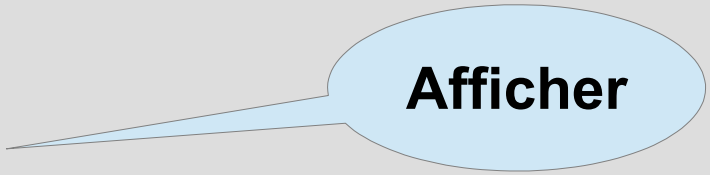


Recursivité



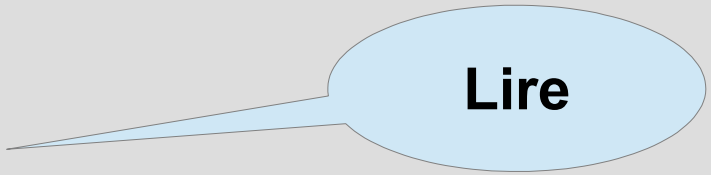
Les interuptions

```
Message DB 'Hello World$'
; instructions
MOV AH,09h
MOV DX,Offset Message
INT 21h
```



Les interuptions

```
; instructions
   MOV AH,0Ah
   MOV DX,Offset A
   INT 21h
```



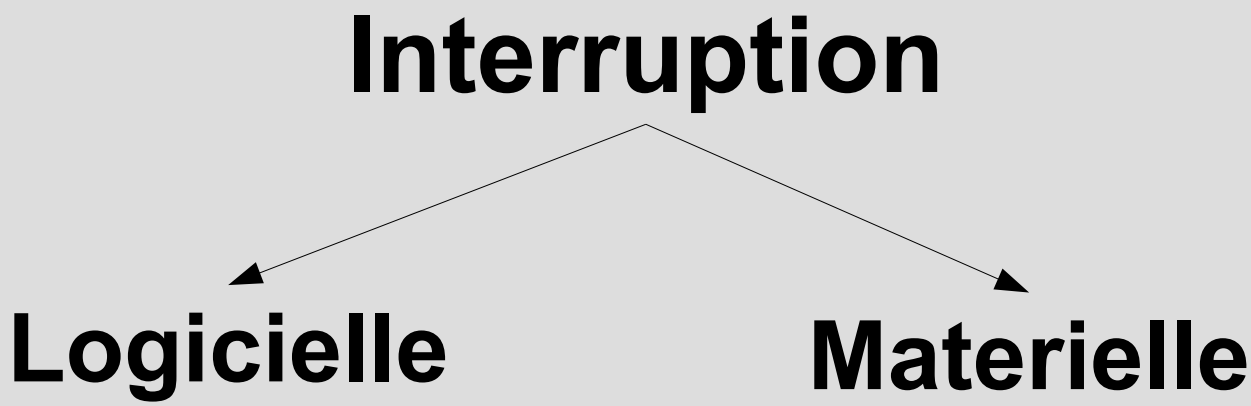
Nous sommes régulièrement confrontés aux interruptions. Certaines peuvent être ignorées (le téléphone qui sonne), d'autres non (un pneu qui crève). En informatique comme dans la vie, l'interruption est généralement prise en charge aussi vite que possible.

On appelle interruption l'événement qui fait qu'un processeur arrête momentanément l'exécution d'une routine pour en exécuter une autre, au plus vite.

Le contraire des interruptions est le polling (scruter).

- déroutements erreur interne du processeur, débordement, division par zéro, page fault, appels systèmes demande d'entrée-sortie par exemple.

Types d'interruption



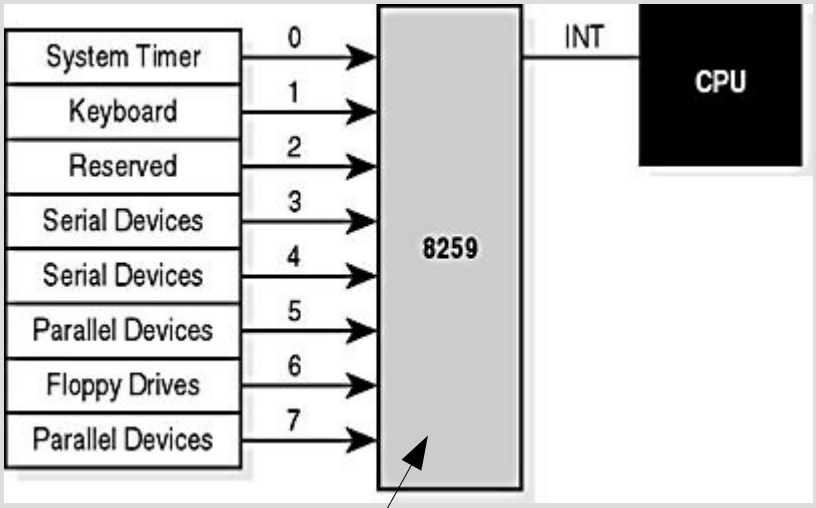
Clavier, Ecran, Sourie, ...

Interruption materielle

- Le controleur d'interruption

Les composants électroniques agissent directement sur une ligne de contrôle : la ligne IRQ " Interrupt Request "

Le contrôleur d'interruptions après avoir répercuté sur la ligne IRQ le signal provenant de l'une de ses entrées doit donner ensuite au processeur le numéro.



Le Intel 8259
Programmable Interrupt Controller (PIC)
https://pdos.csail.mit.edu/6.828/2010/readings/hardware/8259A.pdf

Les contrôleurs d'interruptions peuvent être mis en cascade pour permettre des signaux d'interruptions supplémentaires.

Hiérarchisation des interruptions

- Le processeur dispose d'une seule broche INT pour recevoir les interruptions.
- Que faire : –Si deux interruptions arrivent en même temps ?
- On traite d'abord la plus prioritaire

–Si une interruption survient durant le traitement d'une autre interruption ?

- Elle interrompt le traitement en cours seulement si elle est de priorité supérieure
- On introduit une notion de priorité (niveau) entre interruptions.

Le vecteur d'interruption

Le vecteur d'interruption est une table qui contient les adresses des routines d'interruptions. Les interruptions sont numérotées. Ces numéros servent d'index pour rechercher l'adresse de la routine à exécuter.

Traitement de l'interruption

A l’occurrence d’une interruption, le processeur peut l’ignorer ou l’accepter. Dans le dernier cas, le processeur entreprend une suite d’actions, dont les principales sont :

® Sauvegarde du contexte (Le compteur ordinal, Les registres, Le mot d’état,..)

® Identification de la source de l’interruption

® Exécution de la routine d’interruption

® Restauration du contexte

® Retour au programme interrompu

Exercice

- Écrire le programme qui permet d'afficher une chaine de caractères qui se trouve en mémoire.
- Écrire le programme qui permet de lire une chaine de caractères et de la mettre en mémoire.

Solution

```
.8086
.model small
.stack 10
.data
Message DB 'Hello World$'
.code
start:
MOV AH,09h
MOV DX,Offset Message
INT 21h
end start
```

1

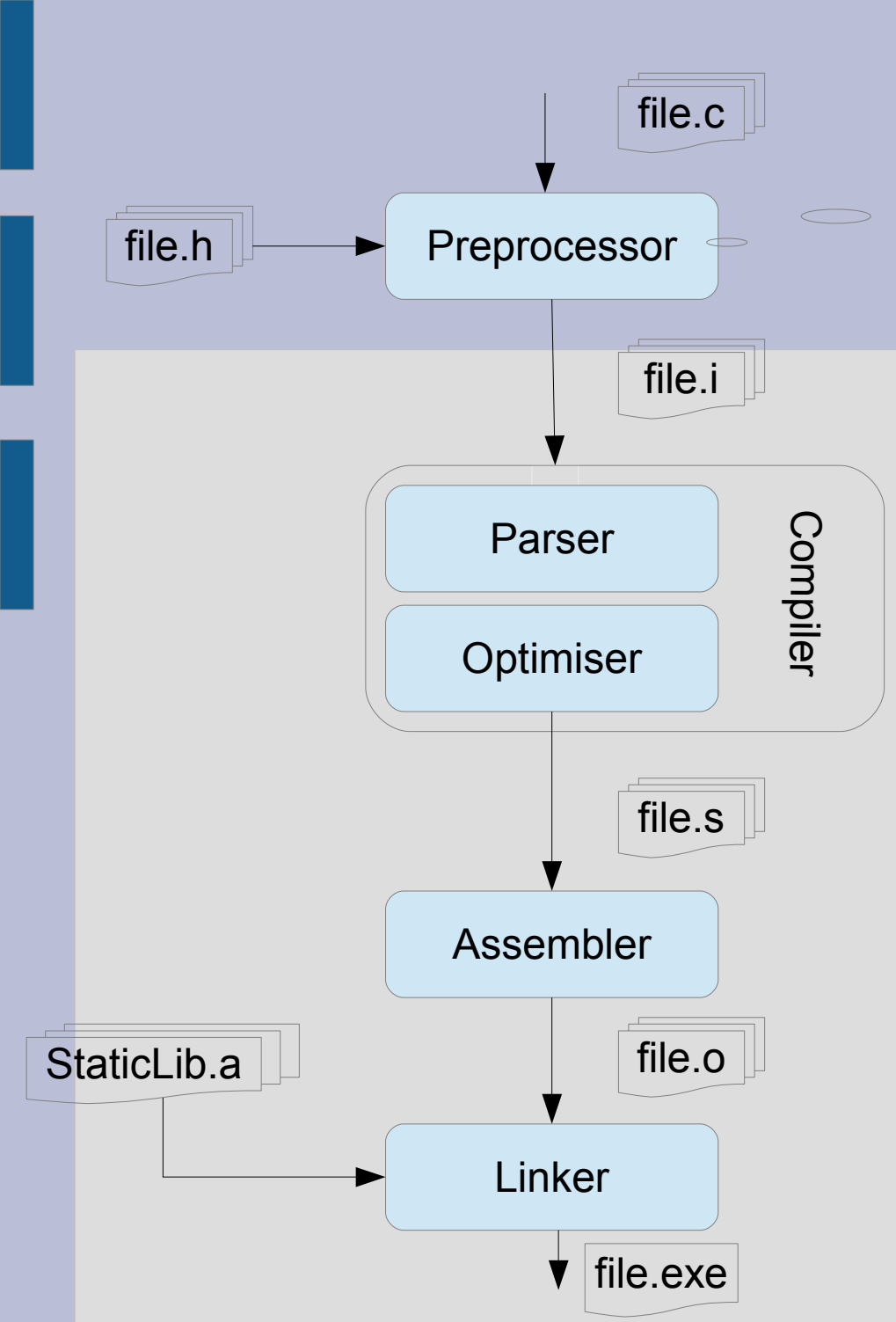
```
.8086
.model small
.stack 10
.data
.code
start:
MOV AH,0Ah
MOV DX,0
INT 21h
end start
```

2

Question

- Comment peut on lire des nombres pour une addition et afficher le résultat (voir la table ascii ci-après)

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL



Compilateur : analyse du programme (lexicale, syntaxique, semantique..), code intermediaire, generation ASM (dependant de l'architecture du CPU), optimisation.

Assembleur : generation code binaire/objet relogeable pour CPU cible.

Editeur de **liens** : liens entre fichiers objets et bibliotheques (statiques et/ou dynamiques) generation code executable

.s fichier assembleur, .a concatenation de fichier .o

Edition des liens

- Jmp et
- Mov al,[adr]
- A DB 'Hello'
- Include
- Call *proc*
- ...

Edition de liens = établir les connexions

-entre les espaces d'adresses des différents modules de code-objet,

-adresses représentées par des références symboliques (symboles définis dans un module et référencés dans un autre).

Edition de liens

***statique : effectuée dans une phase indépendante de l'exécution, et avant le chargement en mémoire**

- l'édition de liens doit physiquement combiner tous les modules de code-objet en un seul,

- pour l'exécution, le code-objet résultant ne doit pas contenir des références externes nonsatisfaites;

*dynamique : effectuée au moment de l'exécution

- le code-objet résultant contient des références externes non-satisfaites - ces références seront traitées pendant l'exécution,

- il n'y a pas la nécessité (mais la possibilité) de physiquement combiner plusieurs modules de code-objet en un seul.

TASM

- Tasm *nomFichier* : assemblage pour obtenir le fichier objet (*nomFichier.o*)
- Tlink *nomFichier* : edition des liens pour obtenir le fichier executable (*nomFichier.exe*)

Language C (gcc)

- gcc -S *nomFichier.c* : génèrele fichier assembleur (*nomFichier.s*)
- gcc -c *nomFichier.c* : génère le fichier objet (*nomFichier.o*)
- gcc -o *nomFichier.c* : génère le fichier executable (*nomFichier.exe*)
- gcc -E *nomFichier.c* : affiche le Preprocessing, ne génère pas de fichier .i