

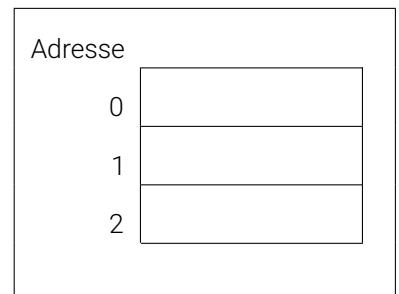
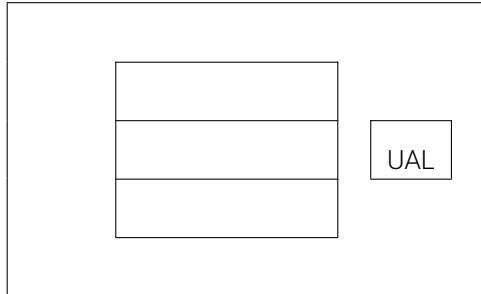
TD1

Architecture et exécution d'un programme

Exercice 1 – Ordinateur simplifié

1. Un ordinateur simplifié dispose d'un compteur ordinal (**IP**), d'un registre d'instruction (**INST**) et d'un registre général (**R0**) et d'une mémoire RAM de 3 octets. Indiquer sur le schéma ci-dessous où se trouvent ces registres, la mémoire RAM et les bus d'adresse, de commande et de données.

Unité centrale de traitement



2. On charge en mémoire le programme suivant :

$$R0 \leftarrow 5$$

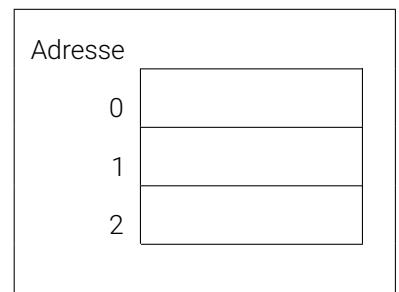
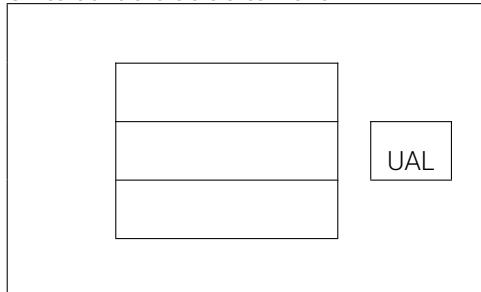
$$R0 \leftarrow R0 + 7$$

On suppose que chaque instruction est codée sur 1 octet.

Compléter le schéma ci-dessus en initialisant IP à 0.

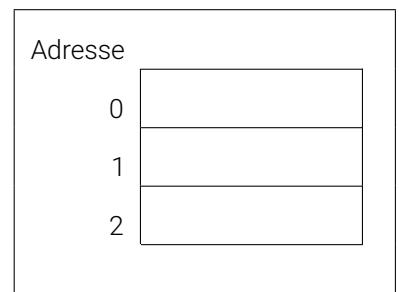
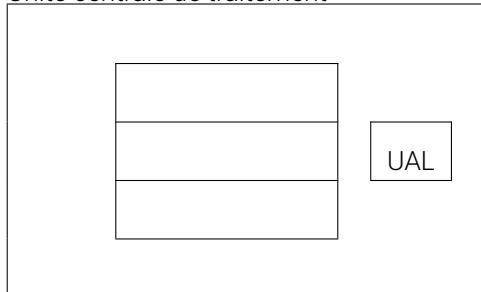
3. Reprendre le schéma précédent en complétant ci-dessous la valeur des bus et les registres dans le schéma ci-dessous après la *lecture* de la première instruction :

Unité centrale de traitement



4. Reprendre le schéma en complétant et en modifiant les valeurs après l'*exécution* de la première instruction :

Unité centrale de traitement



5. Reprendre le schéma en complétant et en modifiant les valeurs après la *lecture* de la deuxième instruction.

6. Reprendre le schéma en complétant et en modifiant les valeurs après l'*exécution* de la deuxième instruction.

Exercice 2 – Une première addition en assembleur

Démarrage :

```

1 assume cs:code
2
3 code SEGMENT
4 debut:
5     mov AL, 27h
6     add AL, 21h
7 code ENDS
8 END debut

```

On a écrit le programme suivant en assembleur pour effectuer l'addition de $27h$ et $21h$:

[CPU 80486]			1=[↑][↓]
cs:0000 B027	mov al,27	ax 0000	c=0
cs:0002 0421	add al,21	bx 0000	z=0
cs:0004 0000	add [bx+sil],al	cx 0000	s=0
cs:0006 0000	add [bx+sil],al	dx 0000	o=0
cs:0008 0000	add [bx+sil],al	si 0000	p=0
cs:000A 0000	add [bx+sil],al	di 0000	a=0
cs:000C 0000	add [bx+sil],al	bp 0000	i=1
cs:000E 0000	add [bx+sil],al	sp 0000	d=0
cs:0010 0000	add [bx+sil],al	ds 48DF	
cs:0012 0000	add [bx+sil],al	es 48DF	
cs:0014 0000	add [bx+sil],al	ss 48EE	
cs:0016 0000	add [bx+sil],al	cs 48EF	
cs:0018 0000	add [bx+sil],al	ip 0000	
ds:0000 CD 20 FF 9F 00 EA FF FF = f Ü			
ds:0008 AD DE E4 01 CA 15 AE 01 iñøùS»			
ds:0010 CA 15 80 02 25 10 93 01 "Søù»ø			
ds:0018 01 01 01 00 02 FF FF FF 0000 0		ss:0002 3A59	
		ss:000000000000	

Après avoir assemblé et lié ce programme, l'exécution pas à pas dans Turbo Debugger a donné le résultat ci-dessous.

1. Quel est le résultat de l'addition ?
2. Quel(s) registre(s) sont utilisés pour effectuer cette addition ?
3. Dans un tableau, représenter la zone mémoire du code avec les valeurs hexadécimales stockant le programme en langage machine.
4. À quel niveau de l'architecture 8086 trouve-t-on la zone mémoire du code ?
5. Comment l'unité centrale de traitement communique-t-elle avec la zone mémoire du code ?
6. Quels sont les codop produits par l'assemblage ? À quelles instructions assebleur correspondent-ils ? Préciser la ou les opérandes.

Après l'exécution de la première instruction :

[CPU 80486]			1=[↑][↓]
cs:0000 B027	mov al,27	ax 0027	c=0
cs:0002 0421	add al,21	bx 0000	z=0
cs:0004 0000	add [bx+sil],al	cx 0000	s=0
cs:0006 0000	add [bx+sil],al	dx 0000	o=0
cs:0008 0000	add [bx+sil],al	si 0000	p=0
cs:000A 0000	add [bx+sil],al	di 0000	a=0
cs:000C 0000	add [bx+sil],al	bp 0000	i=1
cs:000E 0000	add [bx+sil],al	sp 0000	d=0
cs:0010 0000	add [bx+sil],al	ds 48DF	
cs:0012 0000	add [bx+sil],al	es 48DF	
cs:0014 0000	add [bx+sil],al	ss 48EE	
cs:0016 0000	add [bx+sil],al	cs 48EF	
cs:0018 0000	add [bx+sil],al	ip 0002	
ds:0000 CD 20 FF 9F 00 EA FF FF = f Ü			
ds:0008 AD DE E4 01 CA 15 AE 01 iñøùS»			
ds:0010 CA 15 80 02 25 10 93 01 "Søù»ø		ss:0002 3A59	
ds:0018 01 01 01 00 02 FF FF FF 0000 0		ss:000000000000	

Après l'exécution de la deuxième instruction :

[CPU 80486]			1=[↑][↓]
cs:0000 B027	mov al,27	ax 0048	c=0
cs:0002 0421	add al,21	bx 0000	z=0
cs:0004 0000	add [bx+sil],al	cx 0000	s=0
cs:0006 0000	add [bx+sil],al	dx 0000	o=0
cs:0008 0000	add [bx+sil],al	si 0000	p=1
cs:000A 0000	add [bx+sil],al	di 0000	a=0
cs:000C 0000	add [bx+sil],al	bp 0000	i=1
cs:000E 0000	add [bx+sil],al	sp 0000	d=0
cs:0010 0000	add [bx+sil],al	ds 48DF	
cs:0012 0000	add [bx+sil],al	es 48DF	
cs:0014 0000	add [bx+sil],al	ss 48EE	
cs:0016 0000	add [bx+sil],al	cs 48EF	
cs:0018 0000	add [bx+sil],al	ip 0004	
ds:0000 CD 20 FF 9F 00 EA FF FF = f Ü			
ds:0008 AD DE E4 01 CA 15 AE 01 iñøùS»			
ds:0010 CA 15 80 02 25 10 93 01 "Søù»ø		ss:0002 3A59	
ds:0018 01 01 01 00 02 FF FF FF 0000 0		ss:000000000000	

Exercice 3 – Une soustraction en assembleur

On a écrit le programme suivant en assembleur pour effectuer la soustraction de 27h et 21h :

```

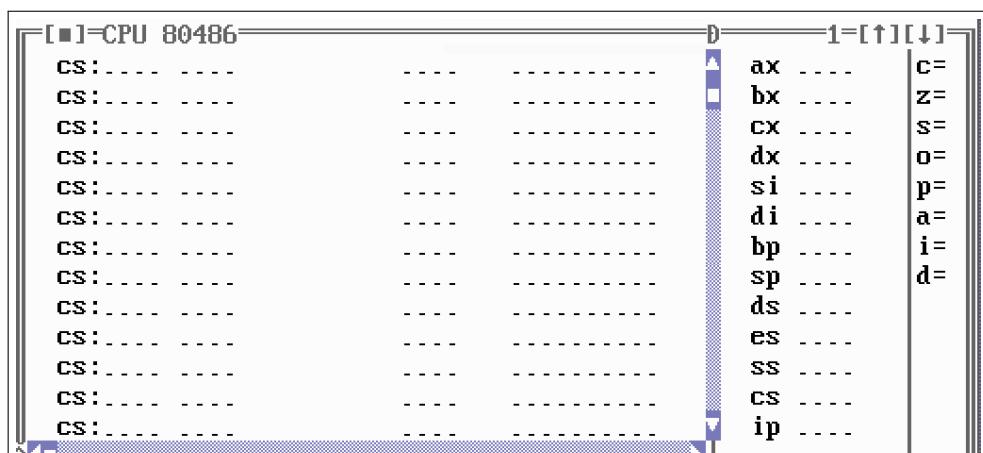
1 assume cs:code
2
3 code SEGMENT
4 debut:
5     mov AL, 27h
6     sub AL, 21h
7 code ENDS
8 END debut

```

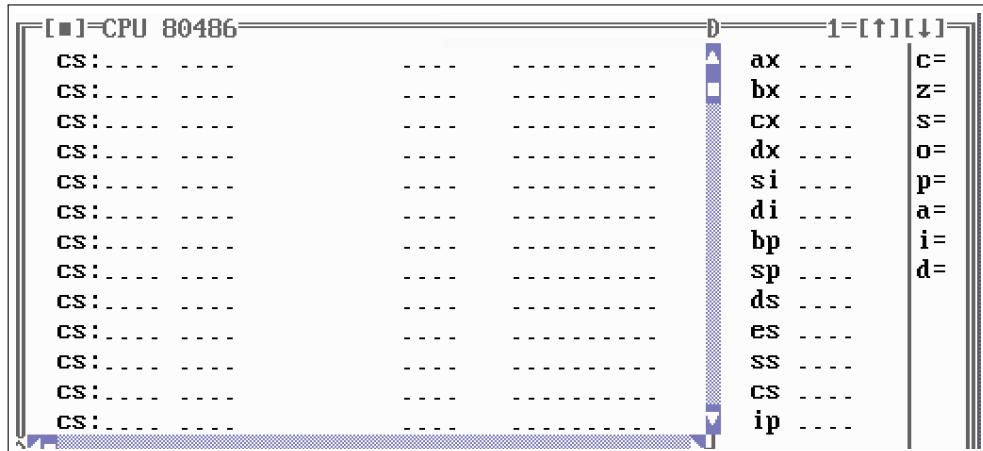
1. Sachant que le codop de l'instruction « `sub AL, ...` » est « `2C` », compléter ci-dessous l'exécution pas à pas dans Turbo Debugger de ce programme (après avoir été assemblé et lié).

2. Représentez la zone mémoire du code dans un tableau.

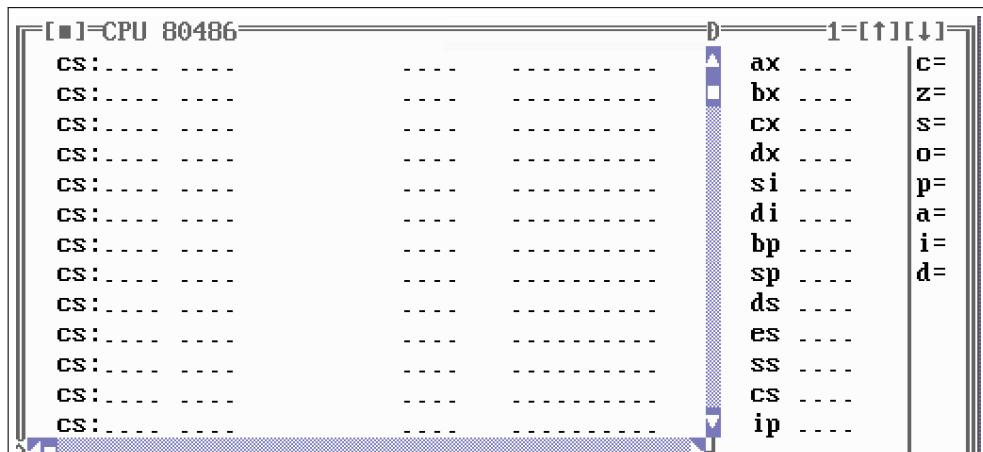
Démarrage :



Après l'exécution de la première instruction :



Après l'exécution de la deuxième instruction :



Exercice 4 – D'autres programmes en assembleur

De la même manière que dans l'exercice précédent, en vous aidant de l'annexe, exécutez pas à pas les programmes suivants et représentez la zone mémoire du code dans un tableau :

1. Une addition avec AX

```

1 assume cs:code
2
3 code SEGMENT
4 debut:
5   mov AX, 1409h
6   add AX, 263h
7 code ENDS
8 END debut

```

2. Une addition de nombres en écriture décimale

```

1 assume cs:code
2
3 code SEGMENT
4 debut:
5   mov AX, 18913
6   add AX, 15616
7 code ENDS
8 END debut

```

3. Une autre addition de nombres en écriture décimale

```

1 assume cs:code
2
3 code SEGMENT
4 debut:
5   mov AX, 142
6   add AX, 151
7 code ENDS
8 END debut

```

4. La même chose (?) avec AL

```

1 assume cs:code
2
3 code SEGMENT
4 debut:
5   mov AL, 142
6   add AL, 151
7 code ENDS
8 END debut

```

5. La même chose (?) avec AL et AX

```

1 assume cs:code
2
3 code SEGMENT
4 debut:
5   mov AX, 0
6   mov AL, 142
7   add AX, 151
8 code ENDS
9 END debut

```

6. La même chose (?)

```

1 assume cs:code
2
3 code SEGMENT
4 debut:
5   xor AX, AX
6   mov AL, 142
7   add AX, 151
8 code ENDS
9 END debut

```

Annexe - Quelques codop

symbole	codop	octets
mov AH, valeur	B4	2
add AH, valeur	80C4	3
sub AH, valeur	80EC	3
mov AL, valeur	B0	2
add AL, valeur	04	2
sub AL, valeur	2C	2
mov AX, valeur	B8	3
add AX, valeur	05	3
sub AX, valeur	2D	3
xor AH, AH	32E4	2
xor AL, AL	32C0	2
xor AX, AX	33C0	2