

## Suite Chap 3 Jeux d'instruction et fin

### 1- Les instructions spéciales :

➤ CMA : complémenter le contenu de l'accumulateur Ac :  $A \longrightarrow \bar{A}$

Exp : MVI A, 55 ; A=55

CMA ; A=AA

➤ CMC ; complémenter le bit carry :  $C \longrightarrow \bar{C}$

Si : C=0  $\Rightarrow C \longleftarrow 1$

Si : C=1  $\Rightarrow C \longleftarrow 0$

Exp : MVI A, FF

ADI A, 15                      A=14 et C=1

CMC

➤ STC ; set bit carry (mettre le bit carry = 1):  $C \longleftarrow 1$

Exp: STC;  $C \longleftarrow 1$

CMC;  $C \longleftarrow 0$

RAL;  $\times 2$ ;  $A \longleftarrow 20$

RAL;  $\times 2$ ;  $A \longleftarrow 40$

RAL;  $\times 2$ ;  $A \longleftarrow 80$

➤ NOP (No Operation): généralement utilisé dans les boucles de temporisation.

Exp : Si : NOP prend un cycle horloge :

Horloge de 1Mhz  $\Rightarrow$  1 cycle horloge = 1 $\mu$ s

Exp : MVI B, FA

Etiquetage NOP

DCR B

JNZ etiquetage

## Les sous-programmes en Assembleur :

- Les sous-programmes sont utilisés pour réduire le nombre de lignes dans un programme principal, ils sont détectés par le programmeur durant la mise en structure et développement du programme principal. Généralement on trouve comme sous-programme :
  - Des temporisations.
  - Des traitements de lecture/écriture.
  - Des traitements de calcul.

### a) Algorithme d'une application en général:

1- Saisie de données :

- Operateur
  - Capteurs (Accele-3D, RC, Temp..)
  - Source de données (Base de données)
- 2- Traitements :
- Trait1
  - Trait2
  - Trait3..
- 3- Présentation des résultats :
- Affichage
  - Stocker les résultats
  - Impression
- 4- Fin.

### b) Les instructions des sous-programmes :

- CALL, Adresse ; instruction pour faire appel à un sous-programme.
- Dans un appel à un sous-programme, le  $\mu P$  sauvegarde l'adresse de la prochaine instruction dans la PILE (mémoire pointé par SP).
- RET : cette instruction définit la fin du sous-programme et le retour au programme principal. L'adresse de retour est récupérée de la PILE. C'est-à-dire dans l'exécution de l'instruction CALL le contenu du registre PC est stocké en mémoire PILE comme suite :

$$M(SP - 2) \longleftarrow PC_L$$

$$M(SP - 1) \longleftarrow PC_H$$

- Dans l'exécution de l'instruction RET, le registre PC sera chargé automatiquement par le contenu de la mémoire PILE comme suite :

$$PC_H \longleftarrow M(SP - 1)$$

$$PC_L \longleftarrow M(SP - 2)$$

Exp : MVI A,25  
 MOV B,A  
 NOP  
 NOP  
 NOP  
 NOP

MVI A, 25  
 MOV B, A  
 MVI A, 6  
 CALL Tempo  
 MVI A, 21  
 MOV C, A

NOP		MVI A, 5
NOP		CALL Tempo
MVI A, 21		MVI A, 15
MOV C, A		MOV D, A
NOP		MVI A, 8
NOP		CALL Tempo
NOP		RST1
NOP	Tempo	NOP
NOP		DCR A
MVI A, 15		JNZ Tempo
MOV D, A		RET
NOP		
NOP		
NOP		
NOP		
NOP		
NOP		
NOP		
NOP		
RST 1		

- Exercice:

- Ecrire un programme qui calcule la somme des valeurs dans l'espace mémoire 8000-8010 , il calcule aussi le nombre de valeurs nul ( $M(i) = 0$ ) , calcule aussi le nombre de valeurs positifs et finalement efface le bloc mémoire 8000-8010 . On range les résultats dans les cases mémoires 8020, 8021, et 8022.

- Note : On utilise des sous-programmes.

- Programme principale :

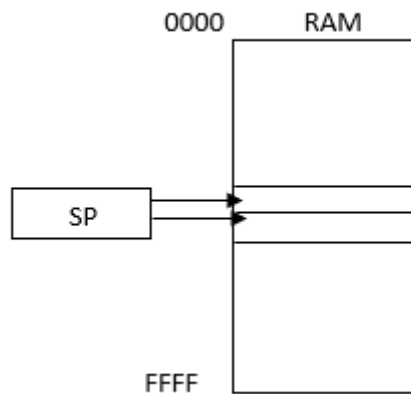
- 1- Initialisation
- 2- Appel SP-Sv (7100)
- 3- Appel SP-VN (7200)
- 4- Appel SP-VP (7300)
- 5- Stocker résultats
- 6- Fin

## 2- Les instructions sur PILE, Contrôle et I/O :

- La pile est une partie de la mémoire RAM que le programmeur choisi en fixant le contenu du registre pointeur de la PILE 'SP' qui pointe à l'adresse du début de la PILE. La PILE est utilisée automatiquement par le  $\mu P$  dans l'exécution de quelques instructions tel que :

Sans conditionnel, CALL, RET, RSTn, PUSH, POP.

La PILE permet de conserver des données dans la mémoire et de les retirer de cette mémoire d'une mémoire LIFO (Last In First Out).

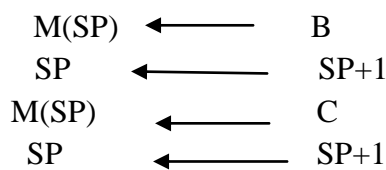


- $RST_n$  : Reset et saut à un sous-programme de traitement de cette conception.

Exp :  $RST\ 1$  : fin de programme à exécuter.

- $PUSH\ reg\text{-}pair$  ; mettre dans la PILE le contenu du registre pair.  
Reg pair : BC, DE, HL.

Exp1 :  $PUSH\ B$

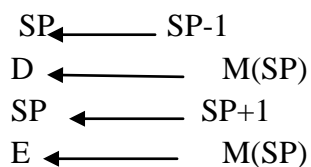


Exp2 :  $LXI\ B, 55AA$

$PUSH\ B$

- $POP\ reg\text{-}pair$  ; retirer de la PILE deux octets consécutives pointée par SP et les ranger dans le registre pair.

Exp :  $POP\ D$

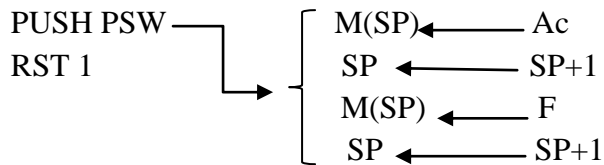


- $PUSH\ PSW$  ; mettre dans la PILE les deux registres actifs Accumulateur et Flag.

Exp1 :  $LXI\ SP, 6800$  ;  $SP \leftarrow 6800$

$MVI\ A, 55$  ;  $Ac \leftarrow 55$

$ADI, F1$



- POP PSW; retirer de la PILE les deux valeurs à ranger dans l'Acc et Flag.

Exp : POP PSW  
RST 1

- XTHL ; changer le contenu de deux cases mémoires consécutives pointé par SP avec le contenu des registres pair H et L.

(L) ← M(SP)  
(H) ← M(SP+1)

- SPHL ; stocker le contenu de HL dans le registre SP.

SP ← HL

Exp : LXI H, 6800      HL ← 6800  
      SPHL              SP ← HL=6800

• Exercice:

- Ecrire un programme qui permute le contenu de 2 cases mémoires en utilisant la PILE.
- Ecrire un programme qui utilise des appels aux sous-programmes afin de voir ce qui se passe au niveau de la PILE.

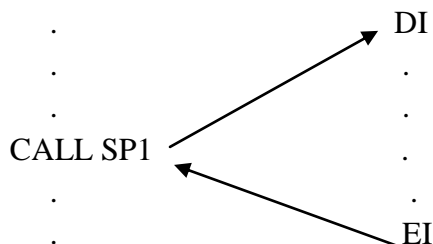
Instruction de contrôle :

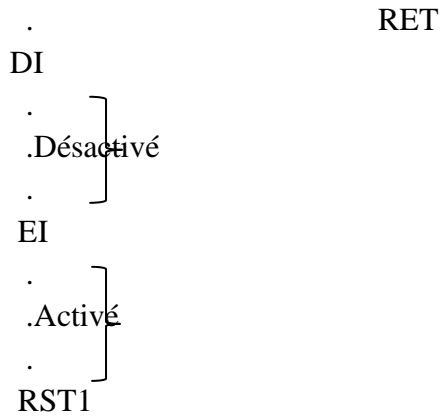
- HLT (Halt) : stopper l'exécution d'un programme.
- EI (Enable Interruption) : activer les interruptions.

Exp : EI si je veux travailler avec les interruptions dans mon programme.

- DI (Desable Interruption) : Désactiver les interruptions

Prog Principal





Les Instructions de transfert de données entre microprocesseur et périphérique.

Tout périphérique lié au système à microprocesseur possède une adresse, exemple dans le cas de la carte SDK-8085 le PIO-0 a une adresse de base : 40H et PIO-1 une adresse de base 50H, le PIO-1 est connecté au 8 LED's via le port A et 8 Interrupteurs via le port B.

Il existe deux instructions pour transférer les données entre périphériques et microprocesseur.

- OUT port-adres ; transférer le contenu de l'accumulateur A vers le port de périphérique dot l'adresse est adrs.

Exp : MVI A, 82

OUT 53 ; programmer les ports du PIO-1 : A en entrée et B en sortie, la programmation se fait dans le registre de contrôle.

MVI A,FF

OUT 50 ; envoi FF sur Port A

Exp : jeux de lumière sur Port A

MVI A,82

OUT 53

Etiquette : MVI A,FF

OUT 50

CALL tempo

MVI A,00

OUT 50

CALL tempo

JMP etiq

RST 1

- IN Port-Adres : lire une donnée du présente dans le Port est range la donnée dans l'accumulateur A.

Acc----- Port-Ardes ;

Exp : lire une donnée sur port B du PIO-1 et range la donnée dans la case mémoire : 8000.

MVI A,82

OUR 53

IN 51

STA 8000

RST 1