Notice d'utilisation du simulateur Z80DT

(Made by the Dutch Open University)

Ahmed ZEBBACHE

Année 2007

Notice d'utilisation du simulateur Z0DT

Introduction :

Développé par la Dutch Open University en 1988, le z80dt est un outil de programmation en assembleur du µPZ80. IL comprend un éditeur, un assembleur (avec la possibilité de générer un listing des fichiers assemblés), ainsi qu'un débuggeur/simulateur, permettant la définition/modification des plages d'adresses pour les ports d'E/S ainsi que pour les données. Une option intéressante, non encore testée actuellement, réside dans la possibilité de transférer le code qui vient d'être assemblé directement dans le MPF-1, par l'intermédiaire d'une liaison RS-232.
Contrairement à ce que son nom pourrait laisser penser, il s'agit donc plus d'un environnement de développement autour du Z80 et permettant une liaison avec le MPF-1 qu'un simulateur stricto sensu.
Dans ce document, nous allons présenter ce simulateur en expliquant son mode d'utilisation.

Présentation :

Le simulateur est téléchargeable à partir de l'Internet à l'adresse suivante :
http://www.z80.info/zip/z80dt.zip

Après l'avoir enregistré sur le disque dure, un clic sur le fichier permet d'obtenir la fenêtre de l'éditeur de texte illustrée sur la figure 1.

La première ligne en haut comprend les commandes du menu principal. On trouve les commandes suivantes :

File, Edit, Assemble, Debug, Load _MPF-1B et Press F10 for main menu

Pour accéder à la fenêtre du menu principal il faut appuyer sur la fonction **F10** du clavier.

Pour accéder à une commande principale il suffit de taper sa **première lettre**.
Ainsi, pour accéder à **E**dit il faut taper **E**, pour accéder à **F**ile, il faut taper **F**, pour accéder à **A**ssemble il faut taper **A**, pour accéder à **D**ebug il faut taper **D**, etc.

Composition de la fenêtre de l'éditeur de texte

Ces explications se rapportent à la fenêtre de la figure 1.

1 : C'est l'espace d'édition du programme. Généralement on commence à écrire le programme à partir de colonne 9 afin de réserver les 8 première s colonnes aux labels (Etiquette). Un commentaire commence toujours par un ' ;' sur la colonne 1. On peut aussi y ajouter des commentaires après l'instruction en mettant en premier lieu un ' ;'
2 : Commande d'accès et de manipulation des fichiers
3 : Passage en mode d'édition de texte.
4 : Assemblage
5 : Déboguage
7 : Retour au menu principal.

Et les commandes en bas de la fenêtre

8 : F1 : Aide
: Ctrl F1 : fournit les commandes d'édition du texte
9 : F2 : sauvegarde du programme dans le répertoire
10 : F3 : Chargement du programme du répertoire vers le simulateur
11 : F5 : Suppression de la bordure de la fenêtre
12 : F7 : Début du blanchissement du texte
13 : F8 : Fin du blanchissement du texte.

Dans les sections suivantes nous allons décrire avec un peu plus de détail les diverses commandes du simulateur.

Fenêtre de l'éditeur du z80dt

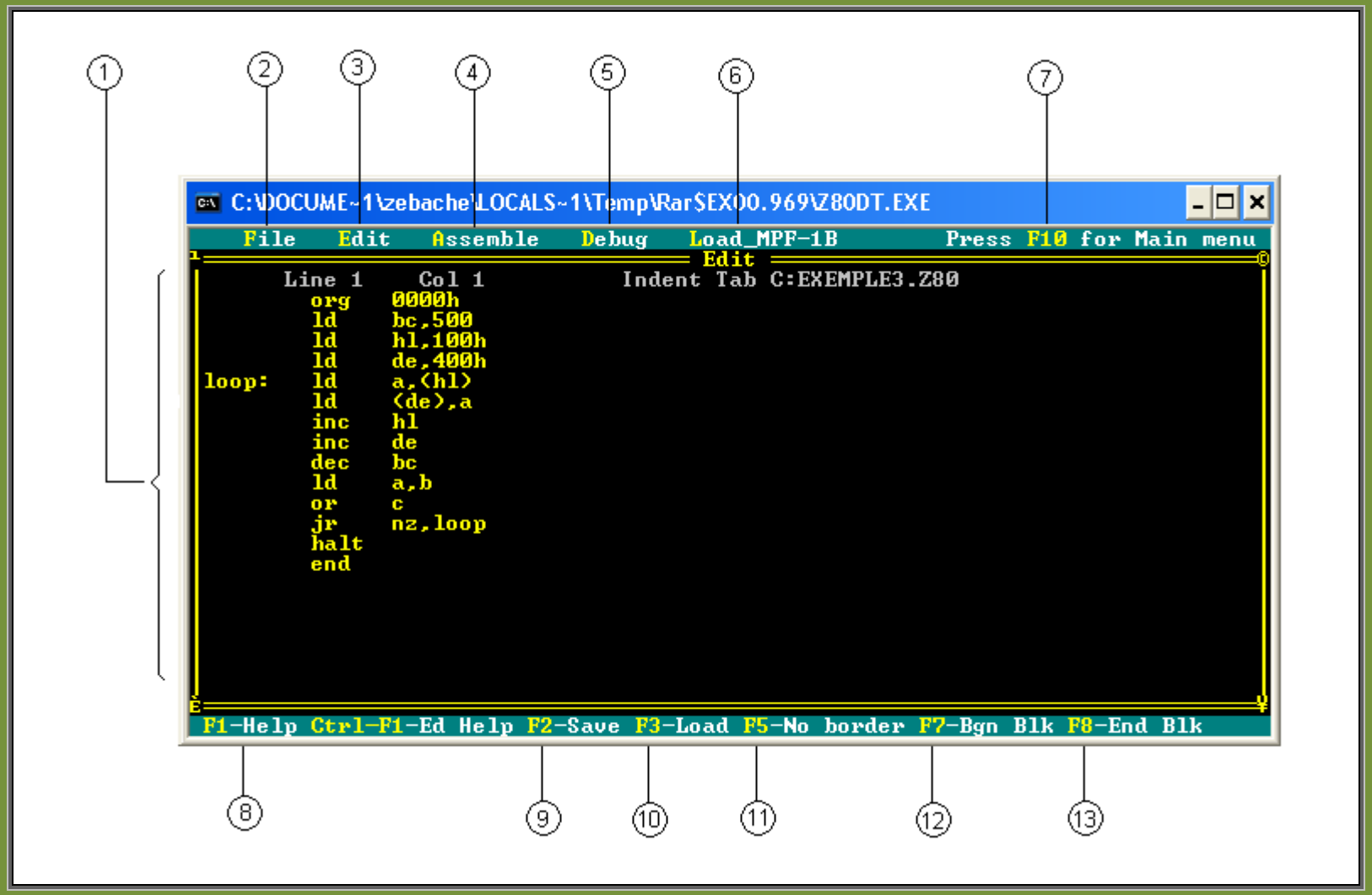
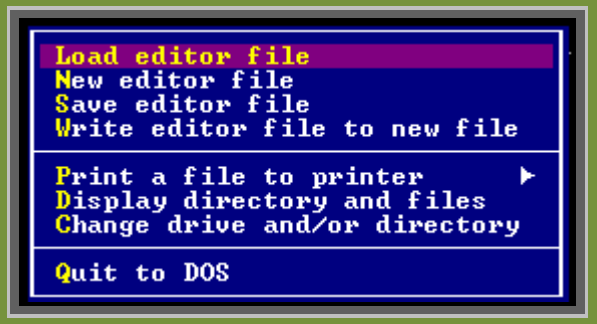


Figure 1 : Fenêtre de l'éditeur du z80dt.

Définition des commandes principales

1. La commande ‘File’ :

La commande ‘**File**’ permet la manipulation du fichier du programme c'est-à-dire : le choix du nom du fichier, sa sauvegarde, son impression, le changement du répertoire et ainsi comment quitter le simulateur.
Lorsqu’on actionne la commande **File** on obtient le sous menu suivant :



On peut accéder à chaque commande du sous-menu en tapant sa **première lettre**. On peut également se déplacer dans le sous-menu à l’aide des flèches ↑↓ du clavier pour sélectionner une commande.

Description des sous menu de la commande ‘File’

Load editor file : permet de charger un fichier dans l’espace d’édition du simulateur. C’est l’espace numéroté (1) sur la figure 1.
Lorsqu’on actionne cette commande on reçoit le message suivant :

A partir de la position de l’astérisque il nous faut taper le nom qu’on souhaite donner du fichier. Le nom du fichier doit comporter au maximum 8 caractères dont le premier est obligatoirement une lettre alphabétique.

A titre d’exemple, si le fichier qu’on veut charger dans l’espace de l’editeur est exemple 1, il suffit de taper ce nom dans la fenêtre du message d’invite :

Ce fichier est généralement sauvegardé dans le répertoire avec l’extension .Z80 c'est-à-dire exemple1.Z80.

New editor file : Cette commande permet de nommer un nouveau fichier. Lorsqu’on actionne cette commande, le simulateur ouvre un nouveau fichier nommé noname.z80.

Save editor file : Cette commande permet de sauvegarder un fichier. Lorsqu’on l’actionne on reçoit le message suivant :

On tape alors le nom qu’on souhaite donner à notre programme (pas plus de 8 caractères). On peut par exemple taper :

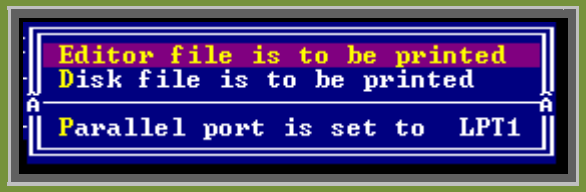
Notre programme sera sauvegardé sous le nom de program1.Z80.

Write file to a new file : Cette commande permet de charger un fichier dans un nouveau fichier. Si on actionne cette commande on reçoit message nous invitons à spécifier le nouveau nom du fichier.

il suffit alors de taper le nouveau nom du fichier ; par exemple Prgram2 et de valider.

Le nom du nouveau programme sera alors sauvegardé sous le nom Program2.Z80.

Print a file to printer : Cette commande permet d'imprimer un fichier sur imprimante. Elle comporte le sous-menu suivant :



La première commande '**Editor file is to be printe**d' permet d'imprimer le fichier sous édition (celui qui se trouve dans la fenêtre de l'éditeur).

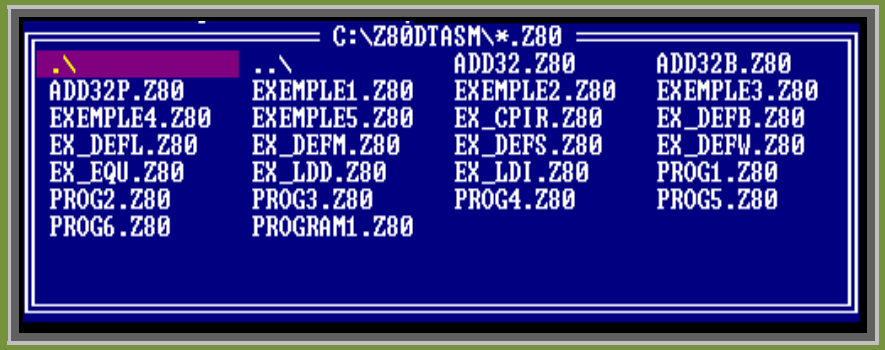
La seconde commande intitulée '**Disk file is to be printed**' imprime un fichier à partir du disk.

La troisième commande '**Parallèle port is set to LPT1**) choisit le port de l'imprimante **LPT1** ou **LPT2**. On passe de LTP1 à LTP2 on actionnant la dite commande

Le simulateur demande le nom du fichier à imprimer qu'il faut spécifier.
Display directory and files : Commande permettant d'afficher le répertoire et ses fichiers. Après le message d'invite taper :

C:\Nom_du_répertoire et validez

Ce dernier affiche son contenu.



Change drive and / or directory : Cette commande vous permet de choisir le répertoire dans lequel vous souhaitez sauvegarder vos fichiers. Après le message d'invite si vous tapez :

L'ordinateur vous ouvre le nouveau répertoire intitulé '**repertoire2'** se trouvant sur le disk C.

Quit to DOS : Cette commande permet tout simplement de quitter le simulateur.

2. La commande 'Edit'

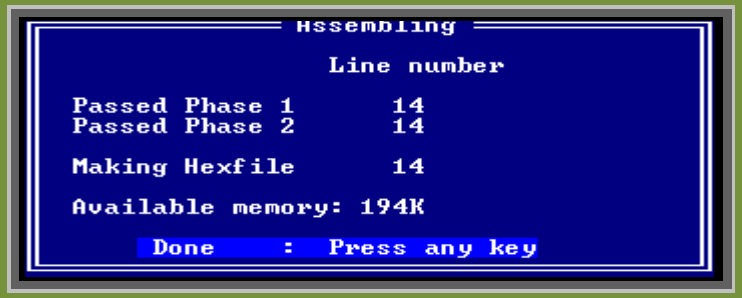
Cette commande vous ramène dans la fenêtre de l'éditeur du texte du simulateur espace d'édition).

3. La commande ‘Assemble’

Cette commande permet l’assemblage d’un programme écrit en assembleur Z80. Il comporte le sous-menu suivant :

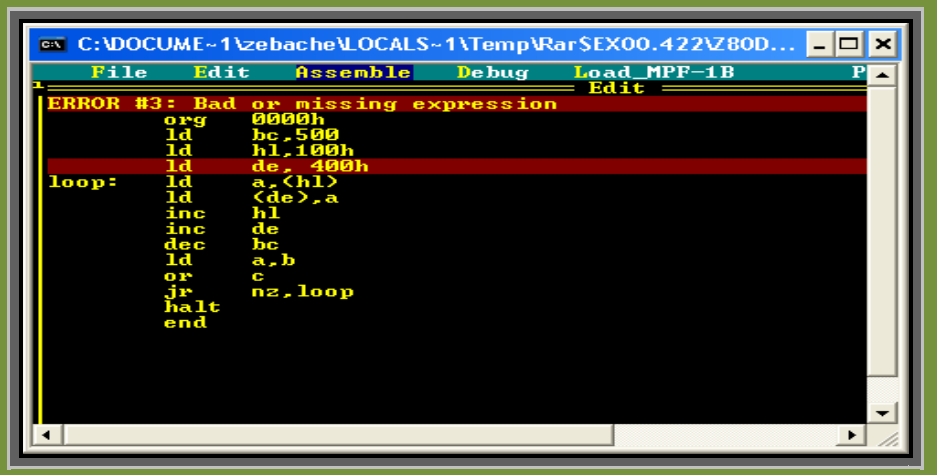
La commande ‘**Assemble Editeur file**’ permet l’assemblage du fichier sous édition. Avant d’assembler le programme choisir l’option ‘**Listing is set to ON** ‘

Cette option permet de sauvegarder le listing du programme avec l’extension .TXT
Si on actionne la commande ‘**Assemble aditor file’** le simulateur vérifie le programme en faisant deux passages et si tout ce passe bien on aura la fenêtre suivante :

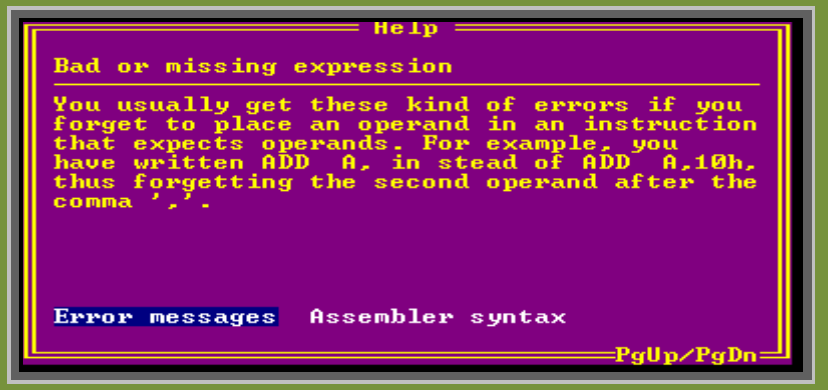


Le nombre 14, dans notre exemple, indique le nombre d’instructions vérifiées. Le nombre 194ko est la taille nécessaire au stockage du programme.
Si le programme comporte des erreurs elles vous seront signalées. A titre d’exemple lors de l’assemblage d’un programme le simulateur nous signale une erreur de syntaxe comme le montre la fenêtre suivante.

Le message d’erreur est imprimé la première ligne sous la forme. Il comporte le numéro de l’erreur et grossièrement sa signification. En outre la ligne erronée est soulignée.
Par exemple, pour le cas de notre programme le message d’erreur est :
ERROR # 3 : Bad or missing expression.



Pour savoir de quoi il s’agit on peut l’utiliser l’aide en actionnant la commande **F1**. On reçoit alors la fenêtre suivante qui nous donne une description précise de l’erreur.



Nous avons laissé un espace entre la virgule et l'opérande ce qui est considéré comme une erreur de syntaxe dans le simulateur Z80dt.

4. La commande 'Debug'

En anglais 'Debug' signifie déboguer c'est-à-dire chercher des bogs (des puces). Les bogs en informatique ce sont les erreurs.
Lorsqu'on actionne cette commande, on accède au sous-menu suivant :

Load code into simulator :

Cette commande permet de charger le programme dans le simulateur. Un fichier du même nom que le fichier initial mais avec l'extension .HEX est créé. Il comporte la traduction du langage assembleur en langage machine. Si vous disposez d'un MPF1-1B (Un MPF1-1B est un microordinateur sous forme de kit conçu pour apprendre à programmer en assembleur) et d'une liaison RS323 pour pouvez télécharger le programme machine directement dans le MPF1-1B.
Pour ce nous concerne cette option n'est actuellement disponible.
Donc lorsque le programme est chargé dans le simulateur, la fenêtre illustrée sur la figure 2 apparaît et on est alors dans le simulateur du Z80dt.

Simulate/Debug : Cette commande permet de simuler et de déboguer. Il nous ramène directement dans la fenêtre du simulateur (Figure 2).

Input port addresses : Cette commande permet d'adresser le port d'entrée en spécifiant une première adresse et/ou une seconde adresse

Output port adresses : Cette commande permet d'adresser le port de sortie en spécifiant une première et/ou une seconde adresse.

Memory address range : Cette commande spécifie l'espace mémoires réservés au programme et aux données. L'espace mémoire total est compris entre **0000H** et **FFFFH**. On spécifie la plus faible adresse avec l'option '**Lowest memory adress'** et la plus haute adresse avec l'option '**Highest memory adress'.**

Initialement les deux valeurs sus citées sont inutilisées à **1800H** et **1FFFH**. Mais on peut les modifier à notre guise grâce à la fenêtre ci-dessus.

Fenêtre du simulateur Z80dt

Lorsque la commande **load code into simulator** est actionnée, une nouvelle fenêtre simulant le z80 avec ses commandes, ses registres, ses flags s'ouvre c'est celle du simulateur illustrée sur al figure 2.

Sur une première fenêtre, figure le programme où chaque instruction et son code sont donnés ainsi que l'emplacement mémoire.

Une deuxième fenêtre comporte tous les registres prime et non prime du Z80 à savoir l'accumulateur A les registres BC, DE et HL ainsi que les registres primes A', BC', DE' et HL'. On trouve également les registres d'index IX, IY, le pointeur de pile SP et le compteur ordinal PC.
On y trouve également le registres des flags F (S Z H P N C) et F' (S' Z' H' P' N' C').

Une autre fenêtre indique les adresses des ports d'entrée et de sortie.

Sous l'espace réservé au programme on y trouve : l'espace réservé aux données de la pile ainsi que la prochaine instruction à exécuter.

En bas de la fenêtre de l'éditeur figure **les commandes de gestion du programme** (F1, F2, F3, F5, F6, F7, F8 et F9).

Fenêtre du simulateur du Z80dt

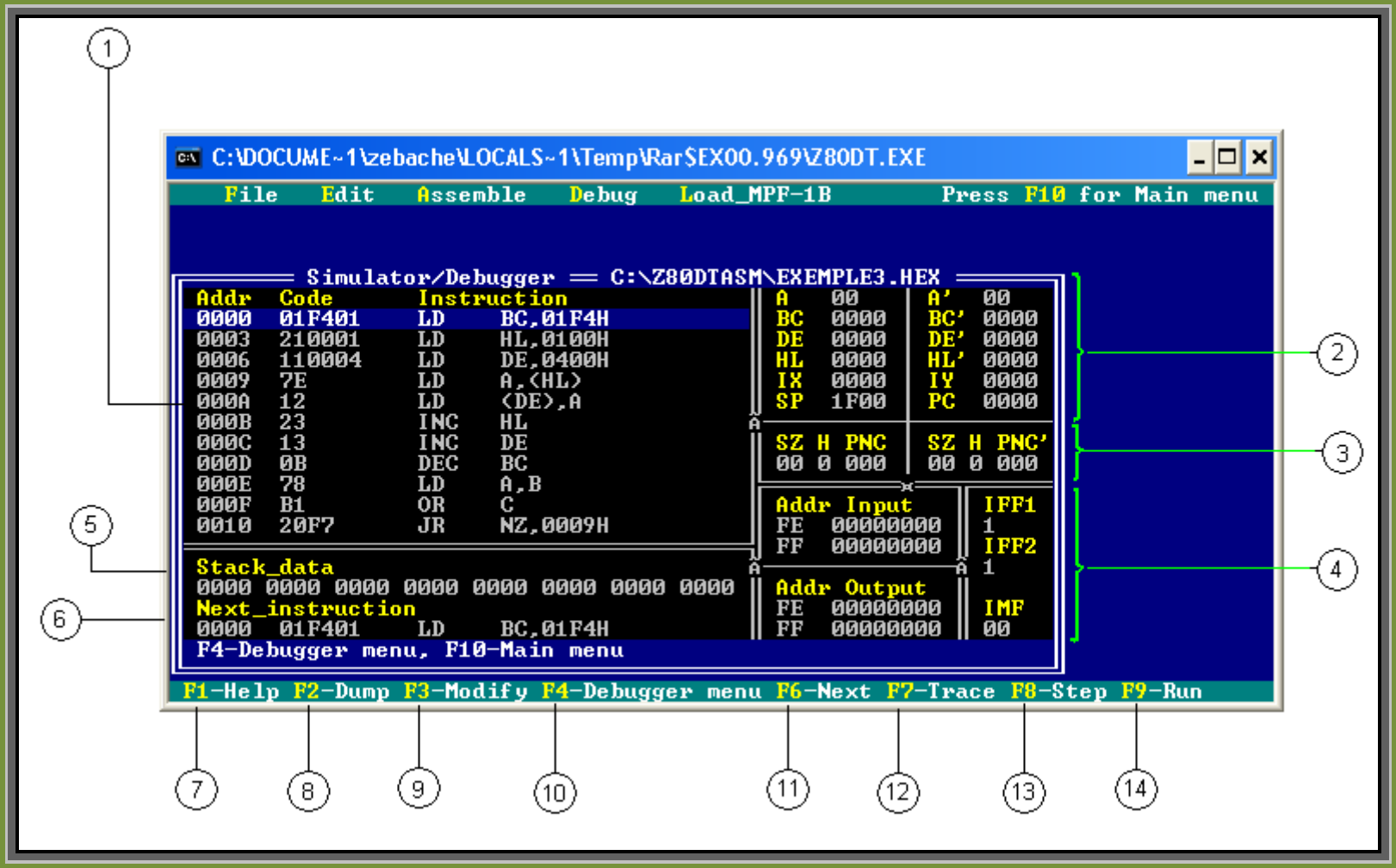


Figure 2 : Fenêtre du débogueur du z80dt.

Description de la fenêtre de la figure 2

1. Espace du programme assemblé (code machine) et assembleur.
2. Fenêtre regroupant les différents registres du Z80
3. Fenêtre regroupant les indicateurs de bits 'Flags'
4. Fenêtre des ports 'entrée et de sortie
5. Fenêtre réservée à la mémoire de données de la pile
6. La prochaine instruction à exécuter.
7. De 7 à 14 ce sont les commandes qui permettent de manipuler le programme (Exécution, déboguage, …etc.

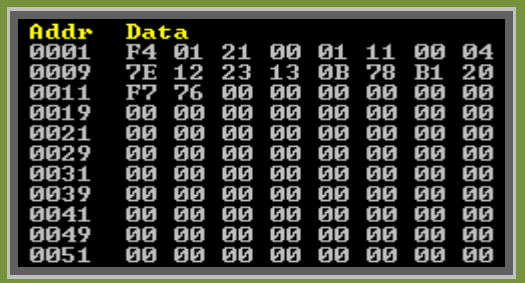
Dans ce qui suit, nous allons faire une description des diverses commandes du simulateur.

Description des commandes de la gestion du programme

La commande F1 (7) **:**
Cette commande fournit une aide**.**

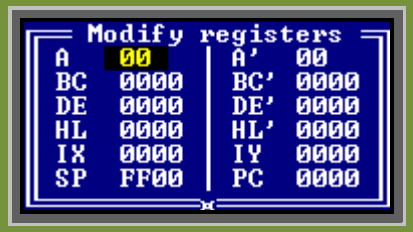
La commande F2 (8) **:**

Cette commande permet d'accéder à l'espace mémoire et de voir le programme implanté sous forme de langage machine ainsi que les données. La mémoire s'échelonne entre 0000H à FFFFH, soit à peu près 64ko. A titre d'exemple, dans la fenêtre ci-dessous, on voit les première case mémoire où le programme est implanté sous forme de code hexadécimal (c'est le langage machine). Cette commande ne permet pas de modifier le contenu des cases mémoires.



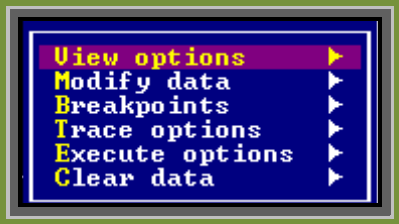
La commande F3 (9) :

Cette commande permet d'accéder aux registres et de modifier leurs contenus. A l'aide des flèches du clavier ← ↓ ↑ → on peut se déplacer d'un registre à l'autre et les modifier facilement.



La commande F4 (10) :

Cette commande assure plusieurs tâches. Lorsqu'on l'actionne on accède au sous-menu suivant :



Description des sous commandes

View options :

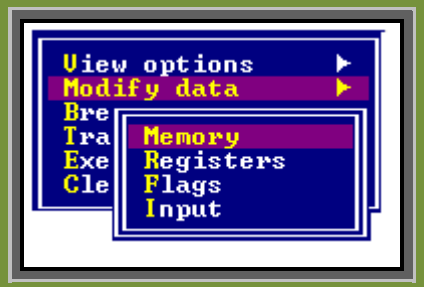
1. Permet d'accéder au programme machine option '**Desassemble**' ou de revenir à la fenêtre précédente option '**DUMP**'

2. Elle permet également d'accéder à l'adresse d'une case mémoire donnée en choisissant l'option **'Go to memory adress'.**

Pour cela il suffit de spécifier l'adresse de la case que vous voulez examiner et de valider.

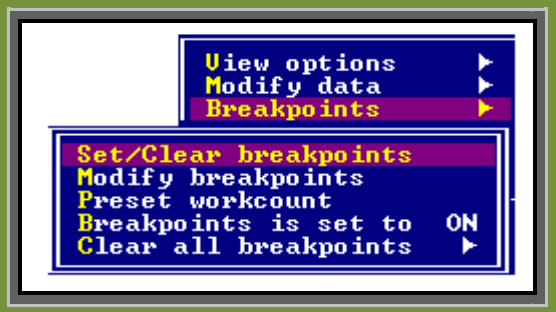
Modify data

Cette commande permet de modifier les données.
Lorsqu'on l'actionne on obtient le sous-menu suivant :

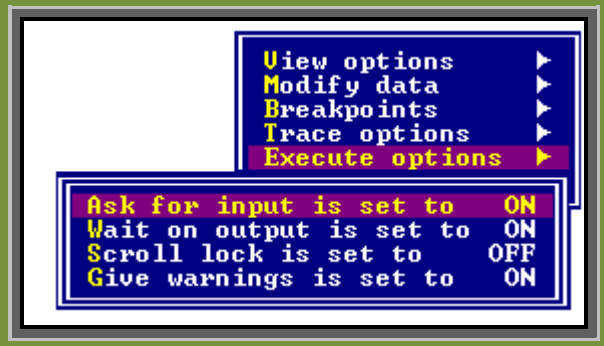


On choisit ce qu'on souhaite modifier (Mémoire, registre, flag ou entrée)

Breakpoints : Cette commande permet de manipuler les interruptions.

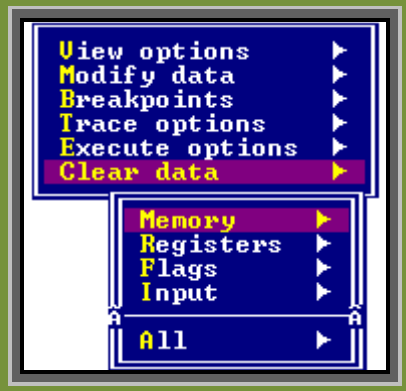


Execute options : permet d'exécuter certaine options (par exemple appel d'entrée) (Attendre une sortie,…etc.



Clear data :

Cette commande permet d'effacer (de remettre à 0) la mémoire **'Memory'**, les registre '**Registers**', les flags **'Flags'** les entrée '**Input**' ou de tout effacer si l'option '**All**' est choisie.



Commande F6 : Cette commande permet d'exécuter l'instruction suivante.
Commande F7 : analogue à RUN (F9) sauf qu'un historique est effectué.
Choisir F4 : **Trace options** est choisir ABOUT (durant) ou AFTER (après) si vous voulez que l'historique soit effectué AVANT APRES OU DURANT l'exécution du programme.

Commande F8 : Cette commande permet d'exécuter le programme pas à pas. On exécute la 1ère instruction ensuite la seconde …etc jusqu'à la fin.

Cette commande est intéressante dans la mesure où elle nous permet de suivre le déroulement du programme.

Commande F9 : Cette commande permet d'exécuter le programme d'un seul coup (exécution rapide).

Comment travailler en plein écran

1. Placer le curseur sur la fenêtre bleu du simulateur et cliquer sur bouton droit de la souris.
2. Sélectionner l'option '**propriétés**' et sélectionner l'option '**plein écran'** et valider en cliquant sur '**OK**' puis de nouveau sur '**OK**'
3. Vous êtes alors en mode plain écran.

Annexe 1

Manuel de programmation du µP Z80

Jeu d'instructions du Z80

Mnémonique	Taille	Format de l'instruction				Nbr cycle	Description de l'opération	S	Z	P/V	C
ADC HL, ss	2	11101101	01ss1010			15	HL+ss+CY dans HL [A←A + ss +Cy]	●	●	●	●
ADC A, r	1	10001 r				4	A+r+CY dans A [A←A + r +Cy]	●	●	●	●
ADC A, n	2	11001110	n			7	A+n+CY dans A [A←A + n +Cy]	●	●	●	●
ADC A, (HL)	1	10001110				7	A+(HL)+CY dans A [A←A + (HL) +Cy]	●	●	●	●
ADC A, (IX+d)	3	11011101	10001110	d		19	A+(IX+d)+CY dans A [A←A + (IX+d) +Cy]	●	●	●	●
ADC A, (IY+d)	3	11111101	10001110	d		19	A+(IY+d)+CY dans A [A←A + (IY+d) +Cy]	●	●	●	●
ADD A, n	2	11000110	n			7	A+n dans A [A←A + n]	●	●	●	●
ADD A, r	1	10000 r				4	A+r dans A [A←A + r]	●	●	●	●
ADD A, (HL)	1	10000110				7	A+(HL) dans A [A←A + (HL)]	●	●	●	●
ADD A, (IX+d)	3	11011101	10000110	d		19	A+(IX+d) dans A [A←A + (IX+d)]	●	●	●	●
ADD A, (IY+d)	3	11111101	10000110	d		19	A+(IY+d) dans A [A←A + (IY+d)]	●	●	●	●
ADD HL, ss	1	00ss1001				11	HL+ss dans HL [HL←HL + ss]				●
ADD IX, pp	2	11011101	00pp1001			15	IX+pp dans IX [IX←IX + pp]				●
ADD IY, rr	2	11111101	00 rr1001			15	IY+rr dans IY [IY←IY + r]				●
AND r	1	10100 r				4	A AND r dans A [A←A et r]	●	●	●	0
AND n	2	11100110	n			7	A AND n dans A [A←A et n]	●	●	●	0
AND (HL)	1	10100110				7	A AND (HL) dans A [A←A et (HL)]	●	●	●	0
AND (IX+d)	3	11011101	10100110	d		19	A AND (IX+d) dans A [A←A et (IX+d)]	●	●	●	0
AND (IY+d)	3	11111101	10100110	d		19	A AND (IY+d) dans A [A←A et (IY+d)]	●	●	●	0
BIT b, r	2	11001011	01 b r			8	Tester le bit b du contenu du registre r	●	●	●	
BIT b, (HL)	2	11001011	01 b 110			12	Tester le bit b du contenu de la case (HL)	●	●	●	

Mnémonique	Taille	Format de l'instruction				Nbr cycle	Description de l'opération	S	Z	P/V	C
BIT b, (IX+d)	4	11011101	11001011	d	01 b 110	20	Tester le bit b du contenu de la case (IX+d)	●	●	●	
BIT b, (IY+d)	4	11111101	11001011	d	01 b 110	20	Tester le bit b du contenu de la case (IY+d)	●	●	●	
CALL cc, nn	3	11 c 100	n	n		17/10	Saut du S/P à la case nn si condition cc est vérifié				
CALL nn	3	11001101	n	n		17/10	Saut Inconditionnel du S/P à la case mémoire nn				
CCF	1	00111111				4	Prendre le Complément du flag C (Carry)				●
CP r	1	10111 r				4	Comparer A avec le contenu du registre r	●	●	●	●
CP n	2	11111110	n			7	Comparer A avec le contenu de la case (IY+d)	●	●	●	●
CP (HL)	2	10111110				7	Comparer A avec le contenu de la case (HL)	●	●	●	●
CP (IX+d)	3	11011101	10111110	d		19	Comparer A avec le contenu de la case (IX+d)	●	●	●	●
CP (IY+d)	3	11111101	10111110	d		19	Comparer A avec le contenu de la case (IY+d)	●	●	●	●
CPD	2	11101101	10101001			16	Comparer bloc non répétitif	●	●	●	
CPDR	2	11101101	10111001			21/16	Comparer bloc répétitif	●	●	●	
CPI	2	11101101	10100001			16	Comparer un bloc non répétitif	●	●	●	
CPIR	2	11101101	10110001			21/16	Comparer un bloc répétitif	●	●	●	
CPL	1	00101111				4	Prendre le Complément A				
DAA	1	00100111				4	Ajustement résultat d'une opération en code BCD	●	●	●	
DEC r	1	00 r 101				4	Décrémenter le registre r de 1 : (r = r – 1)	●	●	●	
DEC (HL)	1	00110101				11	Décrémenter la case (HL) par 1 : (HL)=(HL)-1	●	●	●	
DEC (IX+d)	3	11011101	00110101	d		23	Décrémenter la case (IX+d) par 1 : (IX+d)=(IX+d)-1	●	●	●	
DEC (IY+d)	3	11111101	00110101	d		23	Décrémenter la case (IY+d) par 1 : (IY+d)=(IY+d)-1	●	●	●	
DEC IX	2	11011101	00101011			10	Décrémenter le registre IX par 1 : (IX= IX – 1)				
DEC IY	2	11111101	00101011			10	Décrémenter le registre IY par 1 : (IY= IY – 1)				
DEC ss	1	00ss1011				6	Décrémenter le registre pair ss de 1 : (ss=ss – 1)				
DI	1	11110011				4	Suspendre les interruptions				
DJNZ e	2	00010000	e-2			13/8	Si B ≠0 mettre b=b-1 et Saut à l'étiquette e				

Mnémonique	Taille	Format de l'instruction				Nbr cycle	Description de l'opération	S Z P/V C
EI	1	11111011				4	Autoriser les interruptions	
EX (SP), HL	1	11100011				19	Echanger la case (SP) et registre HL [(SP) ↔HL]	
EX (SP), IX	2	11011101	11100011			23	Echanger la case (SP) et registre IX [(SP) ↔IX]	
EX (SP), IY	2	11111101	11100011			23	Echanger la case (SP) et registre IY [(SP) ↔IY]	
EX AF, AF'	1	00001000				4	Echanger les registres AF et AF' [AF ↔AF']	
EX DE, HL	1	11101011				4	Echanger les registres DE et HL [DE ↔HL]	
EXX	1	11011001				4	(B ↔B'), (C ↔C'), (D ↔D'), (E ↔E'), (H ↔H') et , (L↔L')	
HALT	1	01110110				4	Arrêt du uP	
IM 0	2	11101101	01000110			8	Mettre le mode d'interruption 0	
IM 1	2	11101101	01010110			8	Mettre le mode d'interruption 1	
IM 2	2	11101101	01011110			8	Mettre le mode d'interruption 2	
IN A, (n)	2	11011011	n			11	Charger le registre A à partir du port (n)	
IN r, (C)	2	11101101	01 r 000			12	Charger le registre r à partir de (C)	● ● ●
INC r	1	00 r 100				4	Incrémenter r par 1 [r ←r+1]	● ● ●
INC (HL)	1	00110100				11	Incrémenter le registre (HL) par 1 [HL←HL+1]	● ● ●
INC (IX+d)	3	11011101	00110100	d		23	Incrémenter la case (IX+d) de 1 [(IY+d←IY+d)+1]	● ● ●
INC (IY+d)	3	11111101	00110100	d		23	Incrémenter la case (IY+d) de 1 [(IY+d)←(IY+d)+1]	● ● ●
INC IX	2	11011101	00100011			10	Incrémenter le registre IX par 1 [IX←IX+1]	
INC IY	2	11111101	00100011			10	Incrémenter le registre IY par 1 (IY←IY+1)	
INC ss	1	00ss0011				6	Incrémenter le registre pair ss de 1 (ss←ss+1)	
IND	2	11101101	10101010			16	[HL←C], HL←HL-1, B←B-1	● ● ●
INDR	2	11101101	10111010			21/16	IND jusqu'à B=0	● ● ●
INI	2	11101101	10100010			16	[HL←C], HL←HL+1, B←B-1	● ● ●
INIR	2	11101101	10110010			21/16	INI jusqu'à B=0	● ● ●
JP (HL)	1	11101001				4	Saut à l'adresse pointée par le registre (HL)	

Mnémonique	Taille	Format de l'instruction				Nbr cycle	Description de l'opération	S Z P/V C
JP (IX)	2	11011101	11101001			8	Saut à l'adresse pointée par le registre (IX)	
JP (IY)	2	11111101	11101001			8	Saut à l'adresse pointée par le registre (IY)	
JP cc, nn	3	11 c 010	n	n		10/10	Saut à l'adresse nn si la condition cc est vraie	
JP nn	3	11000011	n	n		10	Saut à l'adresse nn	
JR C, e	2	00111000	e-2			12/7	Saut relatif conditionnel à l'étiquette e si C (Carry)	
JR e	2	00011000	e-2			12	Saut relatif inconditionnel à l'étiquette e	
JR NC, e	2	00110000	e-2			12/7	Saut relatif à l'étiquette e si NC (pas de carry)	
JR NZ, e	2	00100000	e-2			12/7	Saut relatif au label e si NZ (non zéro)	
JR Z, e	2	00101000	e-2			12/7	Saut relatif au label e si Z (Zéro)	
LD A, (BC)	1	00001010				7	Charger la case mémoire (BC) dans le registre A	
LD A, (DE)	1	00011010				7	Charger la case mémoire (DE) dans le registre A	
LD A, I	2	11101101	01010111			9	Charger I dans le registre A	● ● ●
LD A, (nn)	3	00111010	n	n		13	Charger case mémoire (nn) dans le registre A	
LD A, R	2	11101101	01011111			9	Charger R dans le registre A	● ● ●
LD (BC), A	1	00000010				7	Stocker le registre A dans la case mémoire (BC)	
LD (DE), A	1	00010010				7	Stocker le registre A dans la case mémoire (DE)	
LD (HL), n	2	00110110	n			10	Charger valeur n dans case mémoire (HL)	
LD dd, nn	3	00dd0001	n	n		10	Charger la valeur nn dans le registre pair dd	
LD dd, (nn)	4	11101101	01dd1011	n	n	20	Charger la case mémoire (nn) dans le registre dd	
LD HL, (nn)	3	00101010	n	n		20	Charger case mémoire (nn) dans le registre HL	
LD (HL), r	1	01110 r				7	Charger le registre r dans la case mémoire (HL)	
LD I, A	2	11101011	01000111			9	Charger I avec A	
LD IX, (nn)	2	11011101	00100001			20	Charger IX avec case mémoire (nn)	
LD IX, nn	4	1101110	00100000	n	n	14	Charger IX avec la valeur nn	
LD (IX+d), n	4	11011101	00110110	d	n	19	Charger case mémoire (IX+d) avec la valeur n	

Mnémonique	Taille	Format de l'instruction				Nbr cycle	Description de l'opération	S Z P/V C
LD (IX+d), r	3	11011101	01110 r	d		19	Charger la case mémoire (IY+d) avec le registre r	
LD IY, nn	4	11111101	00100001	n	n	14	Charger IY avec la valeur nn	
LD IY, (nn)	4	11111101	00101010	n	n	20	Charger IY avec case mémoire (nn)	
LD (IY+d), n	4	11111101	00110110	d	n	19	Charger la case mémoire (IY+d) avec la valeur n	
LD (IY+d), r	3	11111101	01110 r	d		19	Charger la case mémoire (IY+d) avec le registre r	
LD (nn), A	3	00110010	n	n		13	Stocker le registre A dans la case mémoire (nn)	
LD (nn), dd	4	11101101	01dd0011	n	n	20	Stocker le registre dd dans la case mémoire (nn)	
LD (nn), HL	3	00100010	n	n		16	Stocker reg. HL dans case mémoire (nn)	
LD (nn), IX	4	11011101	00100010	n	n	20	Stocker IX dans la case mémoire (nn)	
LD (nn), IY	4	11111101	00100010	n	n	20	Stocker IY dans la case mémoire (nn)	
LD R, A	2	11101101	01001111			9	Charger R avec A	
LD r, r'	1	01 r r'				4	Charger le registre r avec le registre r'	
LD r, n	2	00 r 110	n			7	Charger le registre r avec la valeur n	
LD r, (HL)	1	00 r 110				7	Charger r avec le contenu de la case (HL)	
LD r, (IX+d)	3	11011101	01 r 110	d		19	Charger r avec le contenu de la case (IX+d)	
LD r, (IY+d)	3	11111101	01 r 110	d		19	Charger r avec le contenu de la case (IY+d)	
LD SP, HL	1	11111001				6	Charger SP avec HL	
LD SP, IX	2	11011101	11111001			10	Charger SP avec IX	
LD SP, IY	2	11111101	11111001			10	Charger SP avec IY	
LDD	2	11101101	10101000			16	[DE]=[HL],HL=HL-1,DE=DE-1,BC=BC-1	●
LDDR	2	11101101	10111000			21/16	LDD jusqu'à BC=0	0
LDI	2	11101101	10100000			16	[DE]= [HL], HL=HL+1,DE=DE+1,BC=BC-1	●
LDIR	2	11101101	10110000			21/16	LDI jusqu'à BC=0	0
NEG	2	11101101	01000100			8	Rendre le registre A négatif (complément à 2)	● ● ● ●
NOP	1	00000000				4	Pas d'opération	

Mnémonique	Taille	Format de l'instruction				Nbr cycle	Description de l'opération	S	Z	P/V	C
OR r	1	10110 r				4	A OR r dans A : [A = A ou r]	●	●	●	0
OR n	2	11110110	n			7	A OR n dans A : [A = A ou n]	●	●	●	0
OR (HL)	1	10110110				7	A OR (HL) dans A : [A = A ou (HL)]	●	●	●	0
OR (IX+d)	3	11011101	10110110	d		19	A OR (IX+d) dans A : [A = A ou (IX+d)]	●	●	●	0
OR (iy+d)	3	11111101	10110110	d		19	A OR (IY+d) dans A : [A = A ou (IY+d)]	●	●	●	0
OTDR	2	11101101	10111011			21/16	Bloc sortie inverse répétitif	●	●	●	
OTIR	2	11101101	10110011			21/16	Bloc sortie direct	●	●	●	
OUT (C) , r	2	11101101	01 r 001			12	Sortie registre r vers (C)				
OUT (n), A	2	11010011	n			11	Sortie du registre A vers port n				
OUTD	2	11101101	10101011			16	Sortie de bloc	●	●	●	
OUTI	2	11101101	10100011			16	Sortie de bloc	●	●	●	
POP IX	2	11011101	11100001			14	Dépiler de la pile dans le registre IX				
POP IY	2	11111101	11100001			14	Dépiler de la pile dans le registre IY				
POP qq	1	11qq0001				10	Dépiler de la pile dans qq				
PUSH IX	2	11011101	11100101			15	Empiler le contenu du registre IX dans la pile				
PUSH IY	2	11111101	11100101			15	Empiler le contenu du registre IY dans la pile				
PUSH qq	1	11qq0101				11	Empiler le contenu du registre pair qq dans la pile				
RES b, r	2	11001011	10 b r			8	Mise à 0 du bit b du registre r				
RES b, (HL)	2	11001011	10 b 110			15	Mise à 0 du bit b de case mémoire d'adresse (HL)				
RES b, (IX+d)	4	11011101	11001011	d	10 b 110	23	Mise à 0 du bit b de case mémoire d'adresse(IX+d)				
RES b, (IY+d)	4	11111101	11001011	d	10 b 110	23	Mise à 0 du bit b de case mémoire d'adresse(IY+d)				
RET	1	11001001				10	Retour de S/P				
RET cc	1	11 cc 00				11/5	Retour de S/P si la condition cc est vérifiée				
RETI	2	11101101	01001101			14	Retour d'une interruption				

Mnémonique	Taille	Format de l'instruction				Nbr cycle	Description de l'opération	S	Z	P/V	C
RETN	2	11101101	01000101			14	Retour d'une interruption. non masquable				
RL r	2	11001011	00010 r			8	Rotation à gauche à travers le Carry de r	●	●	●	●
RL (HL)	2	11001011	00010110			15	Rotation à gauche à travers le Carry de HL	●	●	●	●
RL (IX+d)	4	11011101	11001011	d	00010110	23	Rotation à gauche à travers le Carry de (IX+d)	●	●	●	●
RL (IY+d)	4	11010101	11001011	d	00000110	23	Rotation à gauche à travers le Carry de (IY+d)	●	●	●	●
RLA	1	00010111				4	Rotation à gauche à travers le Carry de A				●
RLC r	2	11001011	00000 r			8	Rotation circulaire à gauche de r	●	●	●	●
RLC (HL)	2	11001011	00000110			15	Rotation circulaire à gauche de (HL)	●	●	●	●
RLC (IX+d)	4	11011101	11001011	d	00000110	23	Rotation circulaire à gauche de (IX+d)	●	●	●	●
RLC (IY+d)	4	11111101	11001011	d	00000110	23	Rotation circulaire à gauche de (IY+d)	●	●	●	●
RLCA	1	00000111				4	Rotation circulaire à gauche de A				●
RLD	2	11101101	01101111			18	Rotation BCD d'1 digit à gauche de (HL)	●	●	●	
RR r	2	11001011	00011 r			8	Rotation à droite à travers le Carry de de r	●	●	●	●
RR (HL)	2	11001011	00011110			15	Rotation à droite à travers le Carry de HL	●	●	●	●
RR (IX+d)	4	11011101	11001011	d	00011110	23	Rotation à droite à travers le Carry de (IX+d)	●	●	●	●
RR (IY+d)	4	00011110	11001011	d	00011110	23	Rotation à droite à travers le Carry de (IY+d)	●	●	●	●
RRA	1	00011111				4	Rotation à droite à travers le Carry de A				●
RRC r	2	11001011	00001 r			8	Rotation circulaire à gauche de r	●	●	●	●
RRC (HL)	2	11001011	00001110			15	Rotation circulaire à gauche de (HL)	●	●	●	●
RRC (IX+d)	4	11011101	11001011	d	00000110	23	Rotation circulaire à gauche de (IX+d)	●	●	●	●
RRC (IY+d)	4	11111101	11001011	d	00000110	23	Rotation circulaire à gauche de (IY+d)	●	●	●	●
RRCA	1	00000111				4	Rotation circulaire à gauche de A				●
RRD	2	11101101	01100111			18	Rotation BCD d'1 digit à gauche de (HL)	●	●	●	
RR r	2	11001011	00011 R			8	Rotation droite à travers le Carry de r	●	●	●	●
RR (HL)	2	11001011	00011110			15	Rotation droite à travers le Carry de (HL)	●	●	●	●

Mnémonique	Taille	Format de l'instruction				Nbr cycle	Description de l'opération	S	Z	P/V	C
RR (IX+d)	4	11011101	11001011	d	00011110	23	Rotation droite à travers le Cy de (IX+d)	●	●	●	●
RR (IY+d)	4	00011110	11001011	d	00011110	23	Rotation droite à travers le Cy de (IY+d)	●	●	●	●
RRA	1	00011111				4	Rotation droite de A à travers le Cy				●
RRC r	2	11001011	00001 r			8	Rotation circulaire à droite de r	●	●	●	●
RRC (HL)	2	11001011	00001110			15	Rotation circulaire à droite de (HL)	●	●	●	●
RRC (IX+d)	4	11011101	11001011	d	00001110	23	Rotation circulaire à droite de (IX+d)	●	●	●	●
RRC (IY+d)	4	11111101	11001011	d	00001110	23	Rotation circulaire à droite de (IY+d)	●	●	●	●
RRCA	1	00001111				4	Rotation circulaire à droite de A				●
RRD	2	11101101	01100111			18	Rotation BCD d'1 digit à droite de (HL)	●	●	●	
RST p	1	11 t 110				11	Retour à la case p				
SBC A, n	2	11011110	n			7	A – n – Cy dans A [A ←A – n – Cy]	●	●	●	●
SBC A, r	1	10011 r				4	A – r – Cy dans A [A ←A – r – Cy]	●	●	●	●
SBC A, (HL)	1	10011110				7	A – (HL) – Cy dans A [A ←A – (HL)– Cy]	●	●	●	●
SBC A, (IX+d)	3	11011101	10011110	d		19	A – (IX+d) – Cy dans A [A ←A –(IX+d) – Cy]	●	●	●	●
SBC A, (IY+d)	3	11111101	10011110	d		19	A –(IY+d)– Cy dans A [A ←A –(IY+d) – Cy]	●	●	●	●
SBC HL, ss	2	11101101	01ss 0010			15	HL – ss – Cy dans HL [A ←A –ss – Cy]	●	●	●	●
SCF	1	00110111				4	Mise à 1 du Carry c=1				1
SET b, (HL)	2	11001011	11 b 110			15	Mise à 1 du bit b de (HL)				
SET b, (IX+d)	4	11011101	11001011	d	11 b 110	23	Mise à 1 du bit b de (HL)				
SET b, (IY+d)	4	11111101	11001011	d	11 b 110	23	Mise à 1 du bit b de (HL)				
SET b, r	2	11001011	11 b r			8	Mise à 1 du bit b de r				
SLA r	2	11001011	00100 r			8	Décalage arithmétique à gauche de r	●	●	●	●
SLA (HL)	2	11001011	00100110			15	Décalage arithmétique à gauche de (HL)	●	●	●	●
SLA (IX+d)	4	11011101	11001011	d	00100110	23	Décalage arithmétique à gauche de (IX+d)	●	●	●	●
SLA (IY+d)	4	11111101	11001011	d	00100110	23	Décalage arithmétique à gauche de (IY+d)	●	●	●	●

Mnémonique	Taille	Format de l'instruction				Nbr cycle	Description de l'opération	S	Z	P/V	C
SLL r	2	11001011	00110 r			8	Décalage logique à gauche de r	●	●	●	●
SLL (HL)	2	11001011	00110110			15	Décalage logique à gauche de (HL)	●	●	●	●
SLL (IX+d)	4	11011101	11001011	d	00110110	23	Décalage logique à gauche de (IX+d)	●	●	●	●
SLL (IY+d)	4	11111101	11001011	d	00110110	23	Décalage logique à gauche de (IY+d)	●	●	●	●
SRA r	2	11001011	00101 r			8	Décalage arithmétique à droite de r	●	●	●	●
SRA (HL)	2	11001011	00101110			15	Décalage arithmétique à droite (HL)	●	●	●	●
SRA (IX+d)	4	11011101	11001011	d	00101110	23	Décalage arithmétique à droite (IX+d)	●	●	●	●
SRA (IY+d)	4	11111101	11001011	d	00101110	23	Décalage arithmétique à droite de (IY+d)	●	●	●	●
SRL r	2	11001011	00111 r			8	Décalage logique à droite de r	●	●	●	●
SRL (HL)	2	11001011	00111110			15	Décalage logique à droite de (HL)	●	●	●	●
SRL (IX+d)	4	11011101	11001011	d	00111110	23	Décalage logique à droite de (IX+d)	●	●	●	●
SRL (IY+d)	4	11111101	11001011	d	00111110	23	Décalage logique à droite de (IY+d)	●	●	●	●
SUB r	1	10010 r				4	A – r dans A [A ← A – r]	●	●	●	●
SUB n	2	11010110	n			7	A – n dans A [A ← A – n]	●	●	●	●
SUB (HL)	1	10010110				7	A – (HL) dans A [A ← A – (HL)]	●	●	●	●
SUB (IX+d)	3	11011101	10010110	d		19	A – (IX+d) dans A [A ← A – (IX+d)]	●	●	●	●
SUB (IY+d)	3	11111101	10010110	d		19	A – (IY+1) dans A [A ← A – (IY+d)]	●	●	●	●
XOR r	1	10101 r				4	A XOR r dans A [A ← A XOR r]	●	●	●	0
XOR n	2	11101110	n			7	A XOR n dans A [A ← A XOR n]	●	●	●	0
XOR (HL)	1	10101110				7	A XOR (HL) dans A [A ← A XOR (HL)]	●	●	●	0
XOR (IX+d)	3	11011101	10101110	d		19	A XOR (IX+d) dans A [A ← A XOR (IX+d)]	●	●	●	0
XOR (IY+d)	3	11111101	10101110	d		19	A XOR (IY+d) dans A [A ← A XOR (IY+d)]	●	●	●	0

Condition des codes

- ● = **Affecté**
- **0** = **Remise à 0**
- 1 = **Remise à 1**
- = **Non Affecté**

Légende :

b : Bit de 0 à 7
c : condition 0=NZ, 1=Z, 2=NC, 3=C, 4=PO, 5=PE, 6=P, 7=M
d : Déplacement indexé de +127 à -128
dd : Registre pair 00=BC, 01=DE, 10=HL, 11=SP
e : Déplacement du saut relatif de +127 à -128.
n : Valeur immédiate ou adresse
pp : Registre pair 00=BC, 01=DE, 10=IX, 11=SP
qq : Registre pair 00=BC, 01=DE, 10=IY, 11=SP
r : Registre 000=B, 001=C, 010=D, 011=E, 100=H, 101=L, 111=A
r' : Pareille à r
rr : BC=00, DE=01, IY=10, SP=11
ss : Registre pair 00=BC, 01=DE, 10=HL, 11=SP
t : Champs de RST Position =tx8**.**

Annexe II

Tableau des caractères ASCII

HEX	**MSD**	0	1	2	3	4	5	6	7
LSD	**BITS**	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
0	**0000**	NUL	DLE	SPACE	0	@	P	`	p
1	**0001**	SOH	DC1	!	1	A	Q	a	q
2	**0010**	STX	DC2	"	2	B	R	b	r
3	**0011**	ETX	DC3	#	3	C	S	c	s
4	**0100**	EOT	DC4	$	4	D	T	d	t
5	**0101**	ENQ	NAK	%	5	E	U	e	u
6	**0110**	ACK	SYN	&	6	F	V	f	v
7	**0111**	BEL	ETB	'	7	G	W	g	w
8	**1000**	BS	CAN	(	8	H	X	h	x
9	**1001**	HT	EM	)	9	I	Y	i	y
A	**1010**	LF	SUB	*	:	J	Z	j	z
B	**1011**	VT	ESC	+	;	K	[	k	{
C	**1100**	FF	FS	,	<	L	\	l	_
D	**1101**	CR	GS	-	=	M	]	m	}
E	**1110**	SO	RS	.	>	N	^	n	~
F	**1111**	SI	US	/	?	O	←	o	DEL

Table des matières