Labo Lo85, utilise le logidules de la boite Lami90 en INF119
La dernière manip demande une EEprom dont le nombre est très limité et ne se trouve que dans les premières boites.

EPFL LAUSANNE - LABORATOIRE DE MICROINFORMATIQUE

Imprimé le 3 mars 1999

LO8085 Le système 8085

\spfdd 12mm-[illegible]:T:LA8AL0";

Objectifs	Compréhension de l'architecture d'un microprocesseur avec bus multiplexé, de son interface mémoire/périphérique, de l'exécution des lectures/écritures en mode pas à pas et des gestions des interruptions
Moyens	Réalisation d'un microordinateur simple basé sur le 8085.
Théorie	Cours "Microprocesseurs I"
Matériel	Boîte logidules, mémoire morte LO85ROM
Durée	4 heures

1. Introduction

Ce laboratoire va permettre de comprendre le fonctionnement en détail des cycles d'un microprocesseur simple, le i8085. Ce laboratoire comprend 3 parties principales:

- Etude des cycles d'accès mémoire et I/O en mode manuel
- Etude des requêtes et accès aux routines d'interruptions
- Réalisation d'un système complet avec EPROM, RAM, I/O et interruptions

2. Le microprocesseur 8085

Le processeur Intel 8085 dispose de 8 lignes de données et d'adresses multiplexées (AD7-AD0), de 8 lignes d'adresses directes A15-A8 et de lignes de contrôle de transferts et d'interruptions.

Le signal ALE (Address Latch Enable) est actif pendant que les adresses de poids faibles sont multiplexées sur le bus AD7-AD0. Les signaux $\overline{RDP}$ (Read Pulse) et $\overline{WRP}$ (Write Pulse) caractérisent le sens de transfert de l'information du processeur. Les cycles sont prolongés tant que READY n'est pas activé, ce qui permet, par exemple, l'exécution en pas à pas.

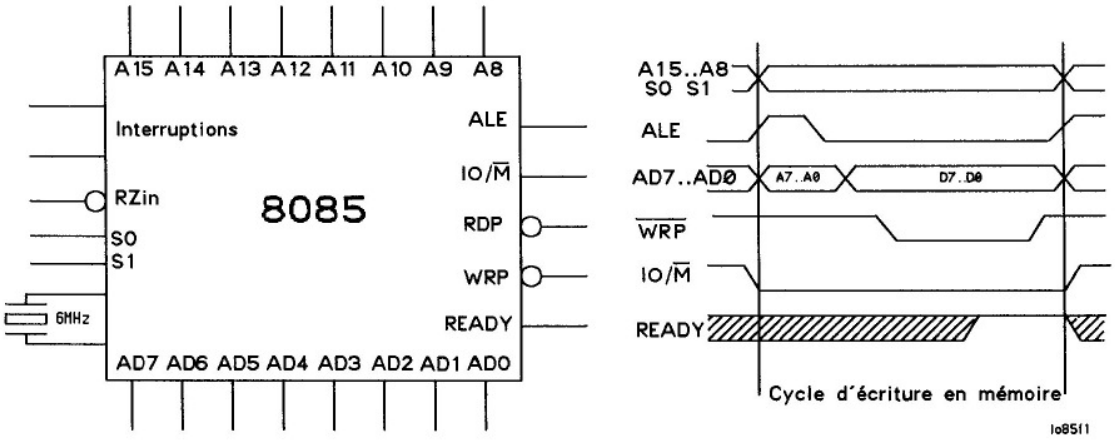


Fig. 1 Signaux principaux du processeur 8085 et diagramme des temps (cycle d'écriture)

Note: Les entrées inutilisées du 8085 doivent avoir un état logique défini. Dans le logidule 8085, des résistances "pull-up" ou "pull-down" assignant une valeur par défaut neutre ont été prévues et sont signalées par un [1], respectivement [0] sur le couvercle du boîtier.

Les cycles possibles sont représentés dans le tableau suivant:

	IO/$\overline{M}$	S1	S0	$\overline{RD}$	$\overline{WR}$	$\overline{Inta}$
Chargement Instruction	0	1	1	0	1	1
Lecture Mémoire	0	1	0	0	1	1
Ecriture Mémoire	0	0	1	1	0	1
Lecture I/O	1	1	0	0	1	1
Ecriture I/O	1	0	1	1	0	1
Quittance Interruption INTA	1	1	1	1	1	0
Bus Idle	0	1	0	1	1	1
Quittance RST, Trap	1	1	1	1	1	1

3. Exécution en mode pas à pas sans mémoire

La 1ère étape de ce laboratoire va permettre de comprendre le fonctionnement des bus d'adresses et de données multiplexés, des signaux de transferts et d'effectuer des cycles de lecture/écriture manuellement en pas-à-pas.

Les adresses sont démultiplexées avec un registre parallèle (PIPO) de type verrou (latch) et affichées (figure 2). Le circuit LS373 est un verrou et garde l'information présente à la fin de l'impulsion ALE. Les codes des instructions et les opérandes sont introduits manuellement à l'aide d'un banc de 8 interrupteurs. Pour éviter tout conflit sur le bus des données (adresses multiplexées, cycles d'écriture), les interrupteurs sont isolés par un passeur (LS541) actif pendant les cycles de lecture seulement. Une logique de pas à pas permet de passer au cycle suivant en activant manuellement le signal READY.

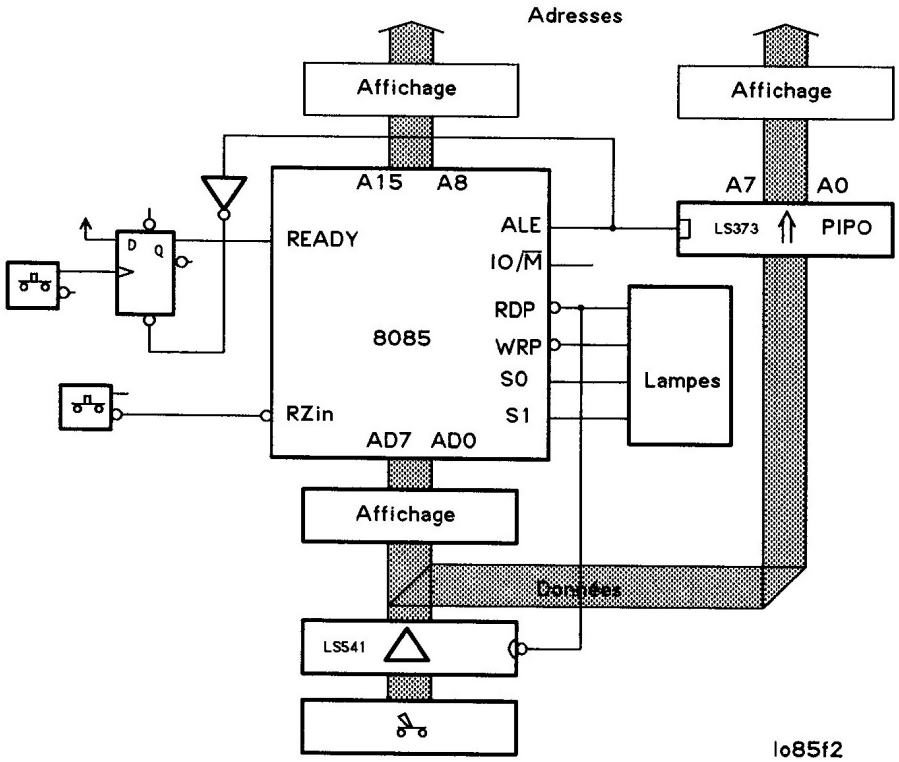


Fig. 2 Schéma pour un système 8085 avec l'opérateur jouant le rôle de mémoire.

Pour que le montage utilise des fils courts et profite des liaisons 8 bits par câble plat, il est recommandé de suivre la disposition de la figure 3.

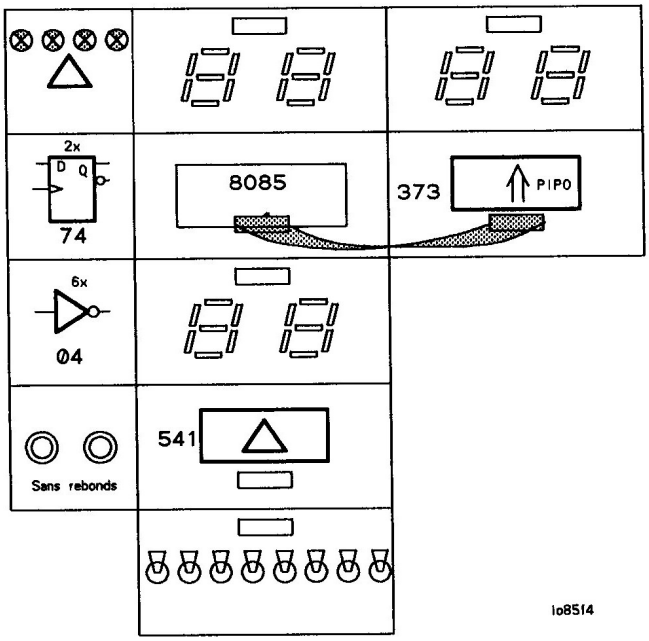


Fig. 3 Disposition conseillée des logidules

3.1 Fonctionnement du processeur

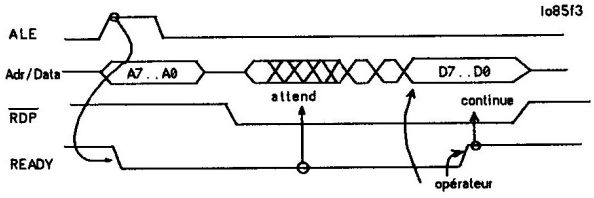
Après activation du signal $\overline{RZin}$, le microprocesseur effectue sa première recherche d'instruction à l'adresse 16'0. Selon le type d'instruction, le processeur lit consécutivement un, deux ou trois octets puis exécute l'instruction. Après exécution, la prochaine instruction est recherchée, puis exécutée, et ainsi de suite.

Un cycle d'accès mémoire s'effectue de la manière suivante :

- le microprocesseur active la ligne ALE et les adresses; les adresses de poids faibles A7..A0 se trouvent sur les lignes multiplexées AD7..AD0
- le microprocesseur désactive la ligne ALE, puis désactive les adresses de poids faibles qui ont été mémorisées dans un verrou sur le flanc descendant de ALE
- s'il s'agit d'un cycle de lecture, le microprocesseur active le signal $\overline{RDP}$, attend que le signal READY soit actif, puis lit les données D7..D0 provenant de la mémoire et finalement désactive $\overline{RDP}$
- s'il s'agit d'un cycle d'écriture, le microprocesseur place les données à écrire sur les lignes AD7..AD0, active le signal $\overline{WRP}$, attend que READY soit actif, puis termine le cycle en désactivant $\overline{WRP}$.

Avec le montage de la figure 2, les données sensées provenir de la mémoire sont fournies par l'utilisateur au moyen des interrupteurs sur les lignes de données.

Les données et adresses sont visualisées sur les modules d'affichage. L'impulsion ALE met la bascule READY à zéro, bloquant le processeur dans son cycle d'attente des données tant que l'opérateur n'a pas agi sur le bouton poussoir (figure 4).



f3

Fig. 4 Diagramme des temps du séquenceur pas-à-pas

Manipulation 1

Montez le dispositif décrit par le schéma de la figure 2 en logidules et testez l'exécution des instructions suivantes :

a)	NOP	code:	00000000
b)	MOVE #data,A	code:	00111110 (16'3E) data
c)	ADD A,A	code:	10000111 (16'87)
d)	MOVE A,16'8000	code:	00110010 (16'32) 0000 0000 (16'00 poids faibles de 16'8000) 1000 0000 (16'80 poids forts de 16'8000)
e)	MOVE #adresse,HL	code:	00100001 (16'21) adresse poids faibles adresse poids forts
f)	JUMP {HL}	code:	11101001 (16'E9)

Vérifiez

- (a) l'incrémentation du compteur ordinal à chaque instruction
- (b) la recherche d'une donnée immédiate 8 bits à l'adresse suivant celle du code de l'instruction
- (c, d) le calcul du double du contenu de A avec affichage en essayant d'écrire dans la position mémoire 16'8000
- (e, f) le saut à l'adresse dont la valeur a été préparée dans HL.

Note: Le processeur 8085 est de type little-endian, les transferts 16 bits sont effectués poids faibles en premier.

Question 1

Répondre sur la feuille annexe

1.1 Combien de cycles faut-il pour exécuter chacune des instructions a) à f) ?

1.2 Le registre A est-il remis à zéro après un Reset ?

1.3 Pourquoi ne peut-on pas remplacer le circuit LS373 (latch, verrou) par un registre dynamique LS374 ?

Manipulation 2

La feuille de codage du 8085 permet de composer des petits programmes et de coder les instructions en langage machine compréhensible par le processeur.

Observez le compteur ordinal (adresse de l'instruction courante du programme en exécution), les lignes de données et les lignes d'adresses, ainsi que les signaux $\overline{RDP}$, $\overline{WRP}$, S0 et S1. Pour le diagramme des temps, l'état observé statiquement est l'état d'attente T_W. Le reste du cycle est conforme à la documentation.

Question 2 Répondre sur la feuille annexe

2.1 Codez et exécutez l'instruction **MOVE.8 A,16'2010**. Tracez le diagramme des temps correspondant à l'exécution de cette instruction après un Reset du μP (signaux ALE, $\overline{RDP}$, $\overline{WRP}$, valeurs S1, S0, adresses mémoire, données).

2.2 Modifiez le schéma de la figure 2 pour que le processeur s'arrête au début de chaque instruction (S1=1, S0=1), et non pas à chaque cycle. Donnez le schéma choisi. Attention, lorsque ALE s'active, S1 et S0 ne sont pas stables!

4. Test des interruptions

Complétons le montage du 8085 en pas à pas sans mémoire pour tester les interruptions (figure 5).

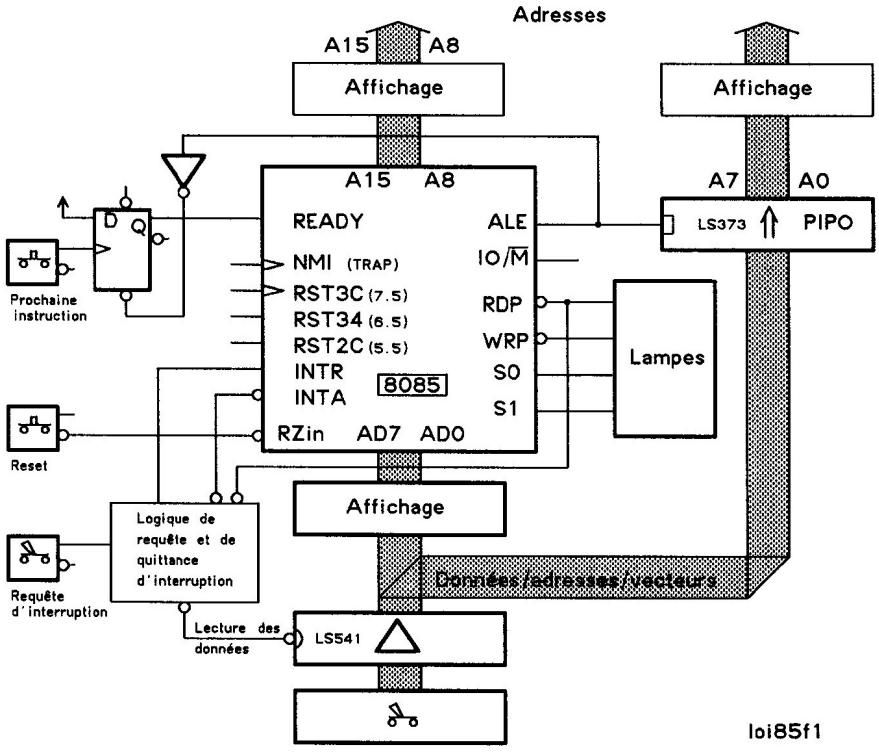


Fig. 5 Schéma du système 8085 pour tester les interruptions

L'entrée d'interruption "vectorisée" s'appelle **INTR**. Elle est active au niveau H. Pour le test, une bascule activée dynamiquement par un interrupteur y sera connectée. L'activation de INTR génère une interruption du processeur à condition qu'il ait préalablement été mis en état *Interrupt On* (instruction ION, code 16'FB). Cette interruption amène le processeur à effectuer un cycle de quittance d'interruption (**INTA**, *INTerrupt Acknowledge*). Lorsque $\overline{INTA}$ s'active le processeur attend sur le bus de données une **instruction** (qui peut comprendre plusieurs octets). En général l'instruction fournie est un **CALL** I (RST I), où I est une des adresses 0, 16'8, 16'10, 16'18, 16'20, 16'28, 16'30 ou 16'38. Cette instruction est codée sur 8 bits (2'11000111 + I). Par (mauvaise) habitude on appelle ceci un "vecteur d'interruption", bien que ce terme soit erroné ici, la donnée fournie n'étant pas réellement un vecteur d'interruption mais bien une **instruction** dans le cas de ce processeur.

Manipulation 3

Montez le système 8085 en pas à pas, selon le schéma de la figure 5. Complétez ce schéma en trouvant une solution pour la logique d'interruption. Les passeurs de lecture de données doivent être activés lors d'un cycle INTA et lecture. Il faut mémoriser une requête qui activera la ligne INTR et lors de la quittance d'interruption, il faut désactiver la requête. Prévoyez des interrupteurs sur les entrées d'interruptions **RSTxx** et une lampe de visualisation sur la sortie **INTA**.

Mettez le processeur dans l'état d'activation des interruptions (**ION**, code 16'FB) et faites tourner le programme dans une boucle ou exécutez des NOP, en pas à pas, cycle par cycle. Créez une demande d'interruption et placez en réponse à INTA le vecteur 2;11*10*0111 correspondant à un "**CALL 16'20**". Observez attentivement les cycles exécutés par le processeur.

Question 3 Répondre sur la feuille annexe

3.1 Donnez le schéma de sélection du passeur de lecture des interrupteurs et de la bascule de demande d'interruption.

3.2 Notez les cycles (adresse, valeur sur le bus de donnée, $\overline{RDP}$, $\overline{WRP}$, INTR, $\overline{INTA}$, S1, S0) exécutés lors de la quittance de l'interruption jusqu'à l'entrée dans la routine d'interruption. Indiquez clairement la nature des cycles ou paires de cycles.

3.3 Quel est le rôle des cycles mémoire suivant la lecture du "vecteur" ?

3.4 Est-ce qu'une interruption active en état IOF est servie par le processeur lorsqu'on active ION ? Que remarque-t-on ?

3.5 Mettez un "vecteur" qui soit une instruction quelconque (par exemple un MOVE #3,A et un CALL 16'1020) et observez la réaction du processeur. Dans quel cas un CALL est intéressant ?

4.1 Interruptions directes RST3C, RST34 et RST2C

Les entrées RST3C, RST34 et RST2C (notées RST7.5, RST6.5, RST5.5 par Intel) sont des entrées d'interruption qui peuvent être masquées selon le contenu du registre interne G. L'instruction SIM (*Set Interrupt Mask*), notée en CALM **SIM A** (code = 16'30) permet de modifier G selon l'état de l'accumulateur A. La partie de G concernant les interruptions est rappelée dans la figure 6.

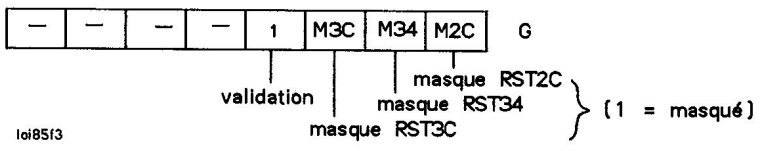


Fig. 6 Registre de masquage des interruptions

Manipulation 4

Testez l'interruption RST2C en cablant un interrupteur sur l'entrée RST2C. Initialisez correctement le processeur.

Question 4 Répondre sur la feuille annexe

4.1 Ecrivez les instructions pour autoriser les interruptions RST2C.

4.2 A partir de quel moment la demande d'interruption RST2C peut-elle être désactivée pour que l'interruption soit reconnue ?

4.3 Ecrivez les instructions pour autoriser les interruptions RST3C.

4.4 A partir de quel moment la demande d'interruption RST3C peut-elle être désactivée pour que l'interruption soit reconnue ?

4.2 Interruption non masquable NMI

L'interruption **NMI** (*Non Maskable Interrupt*) possède la plus haute priorité. Elle est toujours servie immédiatement par le processeur, quel que soit le progamme en cours.

Manipulation 5

Vérifiez l'effet d'une interruption sur NMI et assurez-vous que NMI est servie même en état IOF.

Question 5 Répondre sur la feuille annexe

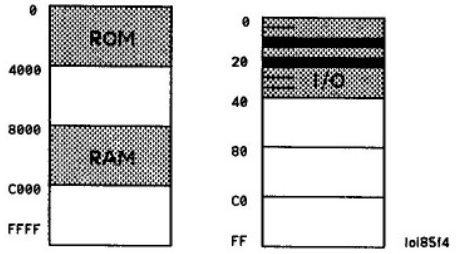
5.1 Quelle est l'adresse de saut de l'interruption NMI ?

5.2 Testez si le processeur est dans l'état ION ou IOF dans la routine NMI (mettre des NOP dans la routine NMI et déclencher une interruption qui a été correctement préparée: est-elle acceptée?).

5. Réalisation d'un système complet à ROM/RAM et à interfaces de sortie

Complétons le montage précédent avec de la ROM (programme), de la RAM (variables, pile), un périphérique d'entrée (interrupteurs sélectionnant le programme à exécuter) et un périphérique de sortie (affichage), ainsi qu'une demande d'interruptions.

Le plan mémoire est donné dans la figure 7. La mémoire est coupée en quatre tranches de 16k. La mémoire morte utilise la première tranche. Avec une EPROM 27128, dont la capacité est de 32k, la moitié de l'EPROM ne sera ainsi pas utilisée. La mémoire RAM de 1k utilise la 3e tranche. Elle est donc placée en 16'8000, et répond aux adresses jusqu'à 16'BFFF, soit une tranche de 16k. Chaque position mémoire peut être trouvée à 16 adresses différentes. La dernière tranche de 16k est libre.



loi8514

Fig. 7 Plan mémoire et périphériques

Le plan des périphériques (adresses 0 à 16'FF, dédoublées sur les poids forts) doit décoder le périphérique d'entrée sur l'adresse 16'10 et le périphérique de sortie, sur l'adresse 16'20. La tranche d'adresse de 0 à 16'3F sélectionne un décodeur sur lequel on trouve les adresse 0, 8, 16'10, 16'18, 16'20,.. 16'38 en lecture et en écriture. L'accès à l'adresse 16'38 permet d'enlever la requête d'interruption.

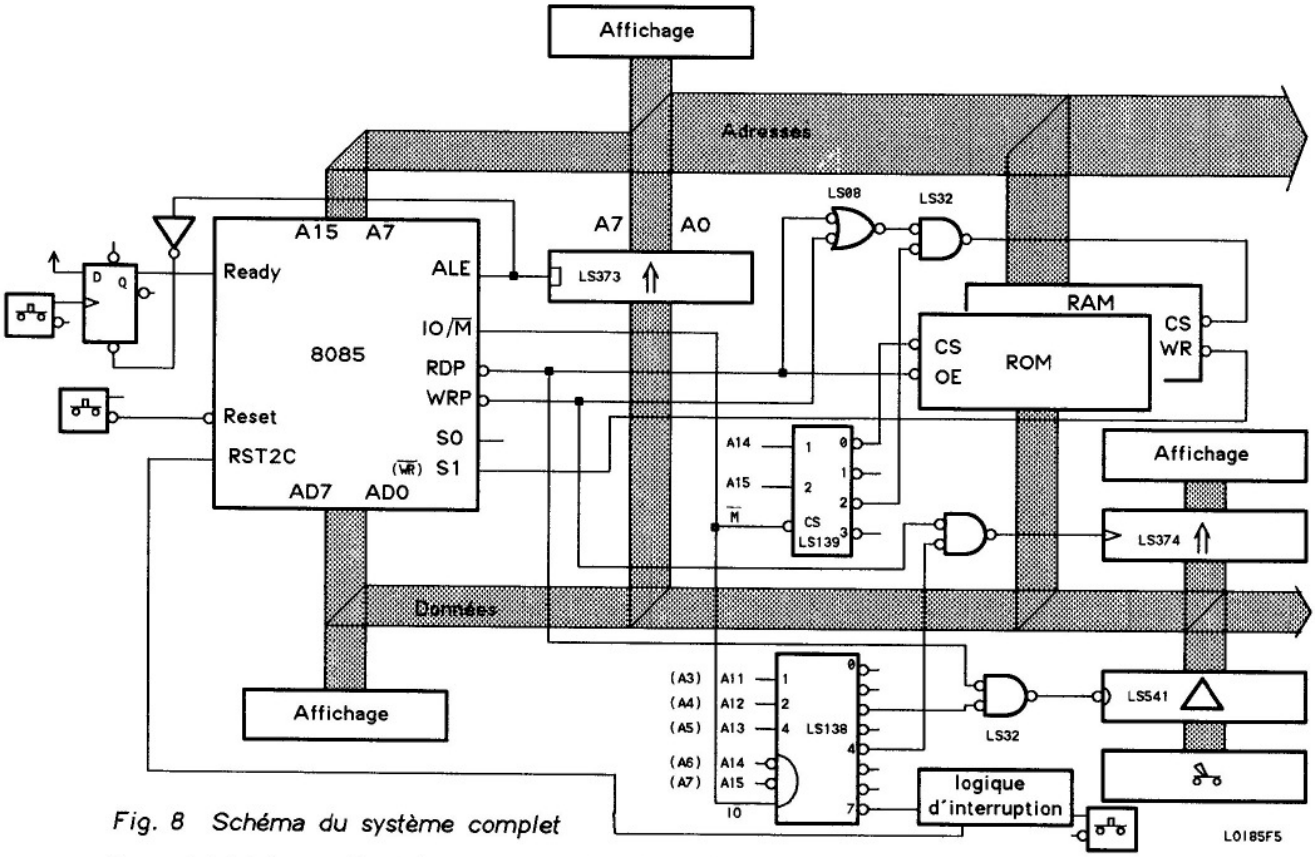


Fig. 8 Schéma du système complet

Le périphérique d'entrée permet de lire 8 interrupteurs. Le périphérique de sortie affiche l'état d'un registre sur 8 lampes. La demande d'interruption vient d'un interrupteur (ou oscillateur réglable) liée à l'entrée d'interruption choisie (RST2C dans ce cas). Une bascule permet de mémoriser la requête qui est à désactiver dans la routine de traitement de l'interruption (accès en 16'38).

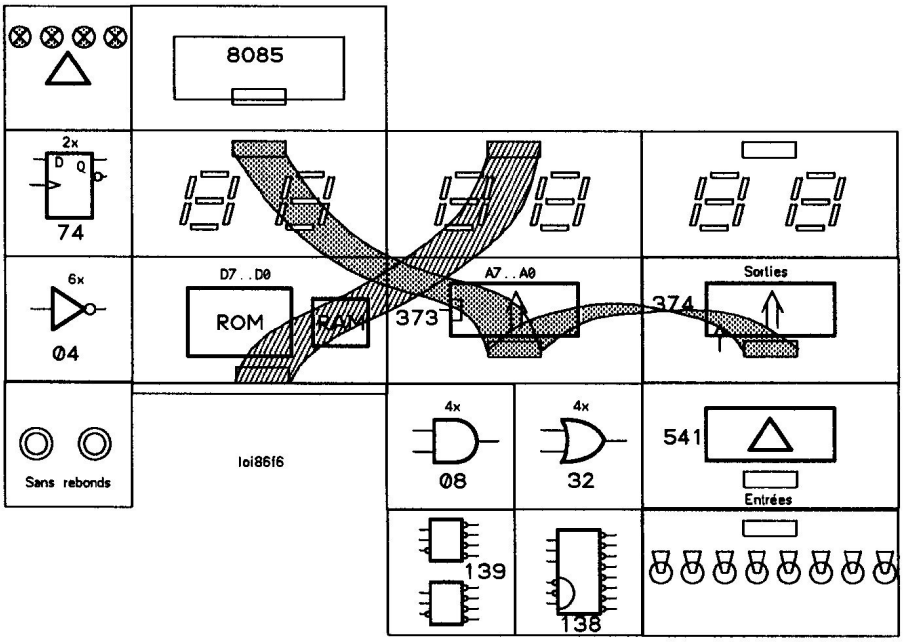


Fig. 9 Topologie des logidules conseillée

5.1 Le programme LO85ROM

Le programme LO85ROM permet de tester un petit système 8085 comportant ROM, RAM et entrées-sorties. Il contient trois programmes, dont l'exécution dépend d'une valeur lue sur les périphériques d'entrée, et deux routines de traitement d'interruptions.

Manipulation 6

Réalisez le montage de la figure 8 et testez le fonctionnement en pas à pas avec la mémoire morte "LO85ROM".

Question 6 Répondre sur la feuille annexe

6.1 Vérifiez que les adresses 8 bits poids forts et 8 bits poids faibles sont identiques pour les accès d'entrée-sortie.

6.2 Certaines RAM ont trois entrée: $\overline{CS}$ (Chip Select), $\overline{OE}$ (Output Enable) et $\overline{WE}$ (Write Enable). Donnez les schémas de sélection dans ce cas (modifications par rapport à la figure 8).

5.2 Test des interruptions

Le programme (C) autorise l'interruption RST2C puis entre dans une boucle de lecture des interrupteurs. La logique d'interruption consiste en une bascule sémaphore activée dynamiquement par l'interruption et désactivée statiquement par le programme ($SERVICE).

Manipulation 7

Générez une interruption RST2C et observer en pas à pas l'appel de la routine d'interruption dans la boucle de lecture des interrupteurs, la remise à zéro de la bascule de demande d'interruption et le retour au programme appelant.

Question 7  Répondre sur la feuille annexe

7.1 Donnez le schéma de la bascule d'interruption

7.2 Que se passe-t-il si la requête n'est pas désactivée dans la routine d'interruption ?

7.3 Que se passe-t-il si une nouvelle demande d'interruption arrive 50ns après que la bascule ait passé à zéro, montrant que l'interruption pendante est servie?

7.4 Faites tourner le processeur à vitesse maximale et générez les demandes d'interruptions avec un oscillateur et augmentez la fréquence. Qu'observez-vous ?

5.3 NMI

Le NMI permet de forcer en tout temps une routine de récupération d'erreur ou de sauvetage (déverminage, chute de tension). Le NMI est sensible à un front montant et la requête doit rester active jusqu'à ce qu'elle soit servie. Dans Lo85ROM, la routine NMI affiche la position SavNMI et l'inverse. Elle ne rétablit pas ION, car elle ne suppose pas que les interruptions sont activées.

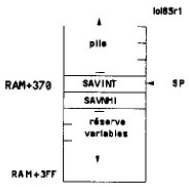
Manipulation 8

Générez des NMI pendant que les tests B et C tournent, et pendant la routine d'interruption.

Question 8  Répondre sur la feuille annexe

8.1 Décrivez vos observations dans les 3 cas.

```
0000                        .TITLE  LO85ROM.ASM
0000                        .PROC   I8080
0000                        .PROCSETI8085
0000
0000  00000020  AFFICH = 16'20          ; adresse 10 I/O en écriture
0000  00000010  SWITCH= 16'10           ; adresse 20 I/O en lecture
0000  00000038  SERVICE= 16'38          ; adresse 38 I/O en écriture
0000  00008000  RAM    = 16'8000        ; adresse RAM (taille 1k)
0000  00008370  STACK  = RAM+16'370     ; pile et variables en fin de mémoire
0000  00008370  SAVINT = STACK          ; il y a encore des positions pour les variables systèmes
0000  00008371  SAVNMI= STACK+1         ;
0000
0000  00000008  EN2C   = 2'1000         ; active l'interruption 2C
0000  00000009  DIS2C  = 2'1001         ; désactive l'interruption 2C
0000
0000  00000000  .LOC   16'0
0000  C34000          JUMP    DEB       ; adresse du programme principal
0024  00000024  .LOC   16'24            ; adresse du programme NMI
0024  C37E00          JUMP    NMI
002C  0000002C  .LOC   16'2C            ; adresse du programme INTERruption
002C  C36F00          JUMP    INTER
0040  00000040  .LOC   16'40
0040  317083    DEB:  MOVE    #STACK,SP
0043
0043  DB10      PROGA: MOVE   $SWITCH,A ; programme (A) > echo direct
0045  D320            MOVE    A,$AFFICH
0047  FE00            COMP    #0,A
0049  CA5300          JUMP,EQ PROGB
004C  3D              DEC     A
004D  CA5B00          JUMP,EQ PROGC
0050  C34300          JUMP    PROGA
0053
0053  3E01      PROGB: MOVE   #1,A      ; programme (B) > lampe se décalant
0055  D320       L1$:  MOVE   A,$AFFICH
0057  07              RL      A
0058  C35500          JUMP    L1$
005B
005B  3E00      PROGC: MOVE   #0,A      ; programme (C) > prépare les interruptions
005D  327083          MOVE    A,SAVINT
0060  3EAA            MOVE    #2'10101010,A
0062  327183          MOVE    A,SAVNMI
0065  3E08            MOVE    #EN2C,A
0067  30              SIM     A       ; SET   A,G
0068  FB              ION
0069  DB10       L1$:  MOVE   $SWITCH,A
006B  00              NOP
006C  C36900          JUMP    L1$
006F
006F                        [INTER] Routine d'interruption
006F  F5        INTER: PUSH   AF
0070  D338            MOVE    A,$SERVICE ; met à zéro la bascule de demande d'interruption
0072  3A7083          MOVE    SAVINT,A
0075  D320            MOVE    A,$AFFICH
0077  3C              INC     A         ; incrémente à chaque interruption
0078  327083          MOVE    A,SAVINT
007B  F1              POP     AF
007C  FB              ION
007D  C9              RET
007E
007E                        [NMI] Routine d'interruption non masquable
007E  F5        NMI:  PUSH    AF
007F  3A7183          MOVE    SAVNMI,A
0082  D320            MOVE    A,$AFFICH
0084  2F              NOT     A         ; complémente à chaque interruption
0085  327183          MOVE    A,SAVNMI
0088  F1              POP     AF
0089  C9              RET
008A
008A                        .END
```



Autres liens soushttps://wiki.epfl.ch/doc-logidules