**Exercice 1 – Décodage d’instructions (désassemblage) [CORRECTION]**

**Travail demandé**

**Identifiez les instructions dans la mémoire puis complétez les registres et les tableaux.**

*Ressource à utiliser :* ***Annexe - Architecture du MC6800 CPU*** ***et instructions utilisées.***

***Objectifs :*** *comprendre comment l’architecture matérielle (modèle de von Neumann) d’un processeur exécute un programme situé en mémoire.* *Désassembler un programme (*[*reverse engineering*](https://www.scip.ch/en/?labs.20211202)*).*

***Première instruction (à la mise sous tension du microprocesseur)***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Opcode**  (Base 16) | **Nombre d’octets (Opcode + Opérande(s))** | **Syntaxe assembleur de l’instruction** | **Opération symbolique** | **Description** | **Remarque**  **1** |
| **FE** | 3 | **LDX** $[add16](http://www.8bit-era.cz/6800.html#data16-desc) | [X] ← [add16] | Charge le registre d’index X avec la donnée sur 16bits (opérande) située aux positions addr16 et addr16+1. | **L**oad**D** **X** |

**PC←PC+3**

|  |  |
| --- | --- |
| **Adresse**  (Base 16) | **Donnée**  (Base 16) |
| **→ 0000** | **FE** |
| 0001 | **00** |
| 0002 | **20** |
| 0003 | 86 |
| 0004 | 00 |
| 0005 | 8B |
| 0006 | 04 |
| 0007 | 09 |
| 0008 | 26 |
| 0009 | FB |
| 000A | B7 |
| 000B | 00 |
| 000C | 22 |
|  |  |
| **0020** | **00** |
| 0021 | **03** |
| 0022 |  |

**Mémoire**

**PC** *(Compteur programme 16bits)*

|  |
| --- |
| **0000** |

**IR** *(Registre d’instruction 8bits)*

|  |
| --- |
| **FE** |

**CPU MC6800 (partiel)**

|  |  |
| --- | --- |
| **X** *(Registre d’index 16bits)* | |
| **0003** | |
| **A**  *(Accumulateur 8bits)* | **B**  *(Accumulateur 8bits)* |
|  |  |

**UAL**

**SR** *(Registre d’état 8bits)*

|  |
| --- |
| Z = 0 |

**Interface Bus**

Horloge

**Unité de contrôle**

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction**  **n°** | **Adresse**  (Base 16) | **Opcode et opérande(s)**  (Base 16) | **Syntaxe assembleur** |
| **1** | **0000** | **FE 00 20** | **LDX $0020** |

***Deuxième instruction***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Opcode**  (Base 16) | **Nombre d’octets (Opcode + Opérande(s))** | **Syntaxe assembleur de l’instruction** | **Opération symbolique** | **Description** | **Remarque**  **2** |
| **86** | 2 | **LDA**[**A**](http://www.8bit-era.cz/6800.html#A-reg) #[data8](http://www.8bit-era.cz/6800.html#data8-desc) | [A] ← data8 | Charge l’accumulateur A avec la donnée sur 8 bits (opérande) qui suit l’opcode. | **L**oad **A**ccumulateur **A** |

**PC←PC+2**

|  |  |
| --- | --- |
| **Adresse**  (Base 16) | **Donnée**  (Base 16) |
| 0000 | FE |
| 0001 | 00 |
| 0002 | 20 |
| **→ 0003** | **86** |
| 0004 | **00** |
| 0005 | 8B |
| 0006 | 04 |
| 0007 | 09 |
| 0008 | 26 |
| 0009 | FB |
| 000A | B7 |
| 000B | 00 |
| 000C | 22 |
|  |  |
| 0020 | 00 |
| 0021 | 03 |
| 0022 |  |

**Mémoire**

**PC** *(Compteur programme 16bits)*

|  |
| --- |
| **0003** |

**IR** *(Registre d’instruction 8bits)*

|  |
| --- |
| **86** |

**CPU MC6800 (partiel)**

|  |  |
| --- | --- |
| **X** *(Registre d’index 16bits)* | |
| **0003** | |
| **A**  *(Accumulateur 8bits)* | **B**  *(Accumulateur 8bits)* |
| **00** |  |

**UAL**

**SR** *(Registre d’état 8bits)*

|  |
| --- |
| Z = 0 |

**Interface Bus**

Horloge

**Unité de contrôle**

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction**  **n°** | **Adresse**  (Base 16) | **Opcode et opérande(s)**  (Base 16) | **Syntaxe assembleur** |
| **2** | **0003** | **86 00** | **LDAA #0** |

**3**

***Troisième instruction***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Opcode**  (Base 16) | **Nombre d’octets (Opcode + Opérande(s))** | **Syntaxe assembleur de l’instruction** | **Opération symbolique** | **Description** | **Remarque** |
| **8B** | 2 | **ADDA** #data8 | [A] ← [A] + data8 | Ajoute au contenu de A, la donnée sur 8 bits (opérande) qui suit l’opcode. | **ADD**ition with **A** |

**PC←PC+2**

|  |  |
| --- | --- |
| **Adresse**  (Base 16) | **Donnée**  (Base 16) |
| 0000 | FE |
| 0001 | 00 |
| 0002 | 20 |
| 0003 | 86 |
| 0004 | 00 |
| **→ 0005** | **8B** |
| 0006 | **04** |
| 0007 | 09 |
| 0008 | 26 |
| 0009 | FB |
| 000A | B7 |
| 000B | 00 |
| 000C | 22 |
|  |  |
| 0020 | 00 |
| 0021 | 03 |
| 0022 |  |

**Mémoire**

**PC** *(Compteur programme 16bits)*

|  |
| --- |
| **0005** |

**IR** *(Registre d’instruction 8bits)*

|  |
| --- |
| **8B** |

**CPU MC6800 (partiel)**

|  |  |
| --- | --- |
| **X** *(Registre d’index 16bits)* | |
| **0003** | |
| **A**  *(Accumulateur 8bits)* | **B**  *(Accumulateur 8bits)* |
| **04** |  |

**UAL**

**SR** *(Registre d’état 8bits)*

|  |
| --- |
| Z = 0 |

**Interface Bus (Adresses et données)**

Horloge

**Unité de contrôle**

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction**  **n°** | **Adresse**  (Base 16) | **Opcode et opérande(s)**  (Base 16) | **Syntaxe assembleur** |
| **3** | **0005** | **8B 04** | **ADDA #4** |

***Quatrième instruction***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Opcode**  (Base 16) | **Nombre d’octets (Opcode + Opérande(s))** | **Syntaxe assembleur de l’instruction** | **Opération symbolique** | **Description** | **Remarque**  **4** |
| **09** | 1 | **DEX** | [X] ← [X] - 1 | Décrémente le registre d’index X. | **DE**crement **X** |

**PC←PC+1**

|  |  |
| --- | --- |
| **Adresse**  (Base 16) | **Donnée**  (Base 16) |
| 0000 | FE |
| 0001 | 00 |
| 0002 | 20 |
| 0003 | 86 |
| 0004 | 00 |
| 0005 | 8B |
| 0006 | 04 |
| **→ 0007** | **09** |
| 0008 | 26 |
| 0009 | FB |
| 000A | B7 |
| 000B | 00 |
| 000C | 22 |
|  |  |
| 0020 | 00 |
| 0021 | 03 |
| 0022 |  |

**Mémoire**

**PC** *(Compteur programme 16bits)*

|  |
| --- |
| **0007** |

**IR** *(Registre d’instruction 8bits)*

|  |
| --- |
| **09** |

**CPU MC6800 (partiel)**

|  |  |
| --- | --- |
| **X** *(Registre d’index 16bits)* | |
| **0002** | |
| **A**  *(Accumulateur 8bits)* | **B**  *(Accumulateur 8bits)* |
| **04** |  |

**UAL**

**SR** *(Registre d’état 8bits)*

|  |
| --- |
| Z = 0 |

**Interface Bus (Adresses et données)**

Horloge

**Unité de contrôle**

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction**  **n°** | **Adresse**  (Base 16) | **Opcode et opérande(s)**  (Base 16) | **Syntaxe assembleur** |
| **4** | **0007** | **09** | **DEX** |

***Cinquième instruction***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Opcode**  (Base 16) | **Nombre d’octets (Opcode + Opérande(s))** | **Syntaxe assembleur de l’instruction** | **Opération symbolique** | **Description** | **Remarque**  **5** |
| **26** | 2 | **BNE** disp | (Z == 0) ? {[[PC](http://www.8bit-era.cz/6800.html#PC-reg)] [←](http://www.8bit-era.cz/6800.html#transfer-desc) [[PC](http://www.8bit-era.cz/6800.html#PC-reg)] + [disp](http://www.8bit-era.cz/6800.html" \l "disp-desc) + 2} | Si la comparaison précédente est fausse, ajoute l’opérande *disp* + 2 au registre PC (compteur programme). **Attention**, la notation est en complément à 2. | **B**ranch if **N**ot **E**qual to zero |

|  |  |
| --- | --- |
| **Adresse**  (Base 16) | **Donnée**  (Base 16) |
| 0000 | FE |
| 0001 | 00 |
| 0002 | 20 |
| 0003 | 86 |
| 0004 | 00 |
| 0005 | 8B |
| 0006 | 04 |
| 0007 | 09 |
| **→ 0008** | **26** |
| 0009 | **FB** |
| 000A | B7 |
| 000B | 00 |
| 000C | 22 |
|  |  |
| 0020 | 00 |
| 0021 | 03 |
| 0022 |  |

**Mémoire**

Si Z = 0

alors

**PC←PC-3**

**PC** *(Compteur programme 16bits)*

|  |
| --- |
| **0008** |

**IR** *(Registre d’instruction 8bits)*

|  |
| --- |
| **26** |

**CPU MC6800 (partiel)**

|  |  |
| --- | --- |
| **X** *(Registre d’index 16bits)* | |
| **0002** | |
| **A**  *(Accumulateur 8bits)* | **B**  *(Accumulateur 8bits)* |
| **04** |  |

**UAL**

**SR** *(Registre d’état 8bits)*

|  |
| --- |
| Z = 0 |

**Interface Bus (Adresses et données)**

Horloge

**Unité de contrôle**

Si Z = 1 alors

**PC←PC + 2**

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction**  **n°** | **Adresse**  (Base 16) | **Opcode et opérande(s)**  (Base 16) | **Syntaxe assembleur** |
| **5** | **0008** | **26 FB** | **BNE -5** |

***Sixième instruction (la fin du bloc de code est atteinte après n parcours de la boucle → vu dans la synthèse)***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Opcode**  (Base 16) | **Nombre d’octets (Opcode + Opérande(s))** | **Syntaxe assembleur de l’instruction** | **Opération symbolique** | **Description** | **Remarque**  **6** |
| **B7** | 3 | **STAA** addr16 | [addr16] [←](http://www.8bit-era.cz/6800.html#transfer-desc) [A] | Sauvegarde le contenu de A à l’adresse addr16. | **ST**ore **A**ccumulator in Memory |

|  |  |
| --- | --- |
| **Adresse**  (Base 16) | **Donnée**  (Base 16) |
| 0000 | FE |
| 0001 | 00 |
| 0002 | 20 |
| 0003 | 86 |
| 0004 | 00 |
| 0005 | 8B |
| 0006 | 04 |
| 0007 | 09 |
| 0008 | 26 |
| 0009 | FB |
| **→ 000A** | **B7** |
| 000B | **00** |
| 000C | **22** |
|  |  |
| 0020 | 00 |
| 0021 | 03 |
| **0022** | **0C** |

**Mémoire**

**PC** *(Compteur programme 16bits)*

|  |
| --- |
| **000A** |

**IR** *(Registre d’instruction 8bits)*

|  |
| --- |
| **B7** |

**CPU 6800 (partiel)**

|  |  |
| --- | --- |
| **X** *(Registre d’index 16bits)* | |
| **0000** | |
| **A**  *(Accumulateur 8bits)* | **B**  *(Accumulateur 8bits)* |
| **0C** |  |

**UAL**

**SR** *(Registre d’état 8bits)*

|  |
| --- |
| Z = 1 |

**Interface Bus (Adresses et données)**

Horloge

**Unité de contrôle**

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction**  **n°** | **Adresse**  (Base 16) | **Opcode et opérande(s)**  (Base 16) | **Syntaxe assembleur** |
| **6** | **000A** | **B7 00 22** | **STAA $0022** |

**Synthèse - Correspondance entre le code source et le code objet du programme**

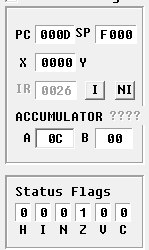


Figure 1- état des registres du CPU à la fin du programme

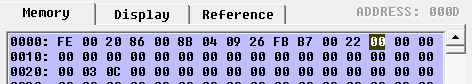


Figure 2 - Code machine [016, C16] et données [2016,2216] dans la mémoire du simulateur à la fin du programme

**1.** Complétez le tableau ci-dessous à l’aide des résultats de l’étude précédente.

**Mémoire programme**

|  |  |
| --- | --- |
| **Adresse**  (Base 16) | **Donnée**  (Base 16) |
| *0000* | FE |
| *0001* | 00 |
| *0002* | 20 |
| *0003* | 86 |
| *0004* | 00 |
| *0005* | 8B |
| *0006* | 04 |
| *0007* | 09 |
| *0008* | 26 |
| *0009* | FB |
| *000A* | B7 |
| *000B* | 00 |
| *000C* | 22 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instruction**  **N°** | **Adresses16** | **Opcode16 et opérande16** | **[label] [opcode] [operand]** | **Opérations symboliques** | **Cycles** |
|  |  |  | .org 0 |  |  |
| **1** | **0000** | **FE 00 20** | **ldx $0020** | [X] ← [002016] | 5 |
| **2** | **0003** | **86 00** | **ldaa #0** | [A] ← 0 | 2 |
| **3** | **0005** | **8B 04** | **loop** **adda #4** | [A] ← [A] + 4 | 2 |
| **4** | **0007** | **09** | **dex** | [X] ← [X] - 1 | 4 |
| **5** | **0008** | **26 FB** | **bne loop** | (Z == 0) ? =>  PC ← PC + FB16 + 2 | 4 |
| **6** | **000A** | **B7 00 22** | **staa $0022** | [002216] ← [A] | 3 |

**2.** Combien de fois la boucle *loop* du programme ci-dessus est-elle parcourue ?

Le programme parcourt 3 fois la boucle *loop* (jusqu’à ce que X = 0)

**3.** Quelle opération est réalisée par ce programme (addition, soustraction, multiplication ou division) ?

Une multiplication : 4 + 4 + 4 = 3 x 4

**4.** Sachant qu’on effectue l’opération **S ← n** \_\_\*\_\_ **k** (0 ≤ S ≤ 255). A l’aide du pseudo-code, complétez l’algorigramme ci-dessous :

**Algorithme** \_\_\_Multiplication\_\_ **Algorigramme**

**Registres Constantes**

X : registre 16 bits k  : entier (0 ≤ k ≤ 16)

**Début**

**Fin**

**X ← n**

**A ← 0**

**A ← A + k**

**X ← X - 1**

**X ≠ 0**

**S ← A**

A : registre 8 bits # contient le résultat S à la fin du programme

**Variables**

n : octet # variable de boucle (0 ≤ n ≤ 15)

S : octet # résultat de l’opération

**début**

X ← n

A ← 0

**faire**

A **←** A + k

X **←** X - 1

**tant que** (X ≠ 0)

S **←** A

**fin**

**5**. Calcul d’un saut

**[[PC](http://www.8bit-era.cz/6800.html" \l "PC-reg)arrivée]**[**←**](http://www.8bit-era.cz/6800.html#transfer-desc)**[[PC](http://www.8bit-era.cz/6800.html" \l "PC-reg)départ] + [disp](http://www.8bit-era.cz/6800.html" \l "disp-desc) + 2**

**PC : Registre Compteur Programme**

**disp** : Déplacement d’adresse signé sur 8 bits

* **bra loop**

[**PC**](http://www.8bit-era.cz/6800.html#PC-reg)**arrivée16 =   8 + FB16 + 2 = 5** (disp+2 = FB16 + 2 = -3)

**5.** Calculez les durées d’exécution minimum et maximum du programme en nombre de cycles ?

Durée = 10 cycles +n\*10 cycles

Pour n = 1 ⇒ Duréemin = 20 cycles

Pour n = 15 ⇒ Duréemax = 160 cycles

# *Annexe - Architecture du microprocesseur MC6800 et instructions utilisées*

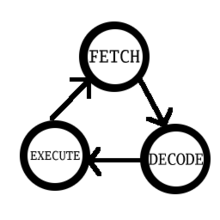


Figure 3 - MC6800(1975)

Nous allons dérouler une séquence d’**instructions** simples de type **langage machine** sur un microprocesseur MC6800. Les instructions sont composées d’**un octe**t, qui représente le **code de l’opération** (**opcode**) à effectuer, éventuellement suivi d’**un ou plusieurs octet(s)** appelé(s) **opérande(s)** de l’opération.

Exemple : la **première instruction** située dans la **mémoire** à partir de l’adresse 000016 est composée de trois octets : **FE**16 ***0020****16*.

**Opcode*****Opérande***



|  |  |
| --- | --- |
| **Adresse**  (Base 16) | **Donnée**  (Base 16) |
| 0000 | FE |
| 0001 | 00 |
| 0002 | 20 |
| 0003 |  |
| 0004 |  |
| 0005 |  |
| 0006 |  |
| 0007 |  |
| 0008 |  |
| 0009 |  |
| 000A |  |
| 000B |  |
| 000C |  |
|  |  |
| 0020 |  |
| 0021 |  |
| 0022 |  |

**Mémoire**

**PC** *(Compteur programme 16bits)*

|  |
| --- |
|  |

**IR** *(Registre d’instruction 8bits)*

|  |
| --- |
|  |

**CPU MC6800 (partiel)**

|  |  |
| --- | --- |
| **X** *(Registre d’index 16bits)* | |
|  | |
| **A**  *(Accumulateur 8bits)* | **B**  *(Accumulateur 8bits)* |
|  |  |

**UAL**

**SR** *(Registre d’état 8bits)*

|  |
| --- |
|  |

**Interface Bus (Adresses et données)**

Horloge

**Unité de contrôle**

**Cycle d’exécution des instructions**

Pour exécuter un programme, l'**unité de contrôle** réalise de façon continue la boucle appelée ***cycle d'exécution d'une instruction***.

1. **FETCH** (chargement). L'*unité de contrôle* récupère l’**opcode** à exécuter à l'adresse mémoire indiquée par le registre **PC** (*program* *counter*) et la place dans le registre **IR** (*instruction register*).
2. **DECODE** (décodage). L’**opcode** contenue dans **IR** est décodée afin de déduire l’action à réaliser et sur quelles données (**opérandes**).
3. **EXECUTE** (exécution). L'instruction est exécutée :

**-** par l'**UAL** (*unité arithmétique et logique*) s'il s'agit d'une opération **arithmétique** ou **logique**,

**-** par l'**unité de contrôle** qui modifie le registre **PC** s'il s'agit d'une **opération de branchement**.

Le tableau ci-dessous est extrait de la source : <https://webge.fr/6800.html> . Il permet de connaître l’organisation d’une instruction, son rôle et sa syntaxe en assembleur MC6800.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Opcode**  (Base 16) | **Nombre d’octets (Opcode + Opérande(s))** | **Syntaxe assembleur de l’instruction** | **Opération symbolique** | **Description** | **Remarque** |
| **8B** | 2 | **ADDA** #data8 | [A] ← [A] + data8 | Ajoute au contenu de A, la donnée sur 8 bits (opérande) qui suit l’opcode. | **ADD**ition with **A** |
| **26** | 2 | **BNE** disp | (Z == 0) ? {[[PC](http://www.8bit-era.cz/6800.html#PC-reg)] [←](http://www.8bit-era.cz/6800.html#transfer-desc) [[PC](http://www.8bit-era.cz/6800.html#PC-reg)] + [disp](http://www.8bit-era.cz/6800.html" \l "disp-desc) + 2} | Si la comparaison précédente est fausse, ajoute l’opérande *disp* + 2 au registre PC (compteur programme). **Attention**, la notation est en complément à 2. | **B**ranch if **N**ot **E**qual to zero |
| **09** | 1 | **DEX** | [X] ← [X] - 1 | Décrémente le registre d’index X | **DE**crement **X** |
| **86** | 2 | **LDA**[**A**](http://www.8bit-era.cz/6800.html#A-reg) #[data8](http://www.8bit-era.cz/6800.html#data8-desc) | [A] ← data8 | Charge l’accumulateur A avec la donnée sur 8 bits (opérande) qui suit l’opcode. | **L**oad **A**ccumulateur **A** |
| **FE** | 3 | **LDX** addr16 | [[X](https://webge.fr/6800.html#X-reg)[(HI)](https://webge.fr/6800.html#HI-desc)] [←](https://webge.fr/6800.html#transfer-desc) [[addr16](https://webge.fr/6800.html#addr16-desc)[(HI)](https://webge.fr/6800.html#HI-desc)], [[X](https://webge.fr/6800.html#X-reg)[(LO)](https://webge.fr/6800.html#LO-desc)] [←](https://webge.fr/6800.html#transfer-desc) [[addr16](https://webge.fr/6800.html#addr16-desc)[(LO)](https://webge.fr/6800.html#LO-desc)] | Charge le registre d’index X avec la donnée sur 16bits (opérande) située aux positions addr16 et addr16+1. | **L**oad**D** **X** |
| **B7** | 3 | **STAA** addr16 | [addr16] [←](http://www.8bit-era.cz/6800.html#transfer-desc) [A] | Sauvegarde le contenu de A à l’adresse addr16. | **ST**ore **A**ccumulator **A** in Memory |

**←** : affectation (la donnée ou l’adresse est transférée dans la direction de la flèche) **[...]** : contenu de ...

**$**: la valeur qui suit est en hexadécimal **disp** : déplacement d’adresse signé sur 8 bits

**#** : la valeur qui suit est une donnée (en l’absence de # c’est une adresse) **addr16** : adresse codée sur 16bits

**data8** : donnée codée sur 8bits