

Fonctionnement des ordinateurs

chapitre VII : processeur

Prof. Xavier Gandibleux

Université de Nantes
Département Informatique – UFR Sciences et Techniques

Année académique 2018-2019

Fonctionnement des ordinateurs



Xavier.Gandibleux@Univ-Nantes.fr 0

Processeur (1/2)

étude détaillée de l'architecture de von Neumann
cycle "Fetch - Decode - Execute"

Processeur

Etude détaillée de l'architecture de von Neumann

Fonctionnement des ordinateurs



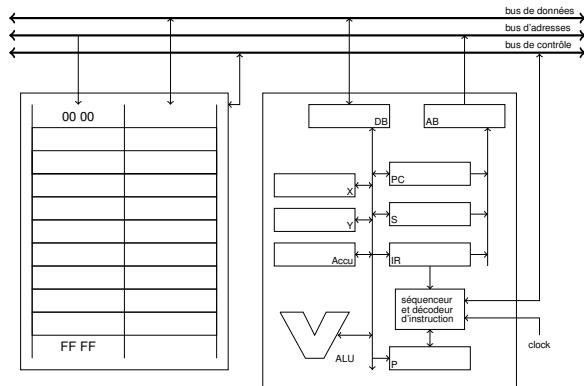
Xavier.Gandibleux@Univ-Nantes.fr 2

Architecture de von Neumann

Les ordinateurs actuels ont une architecture caractérisée en 1945 par John von Neumann :

- ▶ mémoire (données et programme)
- ▶ unité arithmétique et logique (UAL)
- ▶ unité de contrôle (séquençage des opérations)
- ▶ dispositifs d'entrée-sortie

Architecture de von Neumann

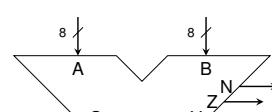


Fonctionnement des ordinateurs



Xavier.Gandibleux@Univ-Nantes.fr 4

Unité arithmétique et logique (UAL[FR]/ALU[GB])



Exemple d'une UAL

Op ₃	Op ₂	Op ₁	Op ₀	Opération
0	0	0	0	S = A
0	0	0	1	S = A ou B
0	0	1	0	S = non (A ou B)
0	0	1	1	S = non (A et B)
0	1	0	0	S = A et B
0	1	0	1	S = B
0	1	1	0	S = non B
0	1	1	1	S = A xor B
1	0	0	0	S = A - B
1	0	0	1	S = A + B
1	0	1	0	S = A * A
1	0	1	1	S = A - 1
1	1	0	0	S = A + 1
1	1	0	1	S = FF
1	1	1	0	S = 0
1	1	1	1	S = non A

Fonctionnement des ordinateurs



Xavier.Gandibleux@Univ-Nantes.fr 5

Registres

Un processeur utilise des registres pour stocker les arguments et les résultats des opérations élémentaires.

L'accès aux registres est plus rapide que l'accès à la mémoire.

Registres généraux ; exemples :

- ▶ A : tout usage (accumulator)
- ▶ X : tout usage et indexation (registre d'index)
- ▶ Y : tout usage et indexation (registre d'index)

Registres spécialisés ; exemples :

- ▶ PC : compteur de programme (Program Counter)
- ▶ IR : registre d'instruction (Instruction Register)
- ▶ S : pointeur de pile (Stack Pointer)
- ▶ P : registre d'état (Processor Status)

Registres

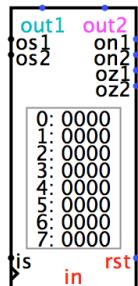
Registre d'état est un ensemble de drapeaux (Flags) donnant un tableau de bord du processeur. Exemples :

- ▶ N : Negative Flag
N est activé si le résultat de la dernière opération présente le MSB à 1
- ▶ Z : Zero Flag
Z est activé si le résultat de la dernière opération vaut zéro
- ▶ V : Overflow Flag
V est activé à l'issue d'opérations arithmétiques si le résultat a donné un résultat de complément à 2 invalide (ex : $64 + 64 \Rightarrow -128$)
- ▶ C : Carry Flag
C est activé si le résultat de la dernière opération a causé un dépassement de capacité du MSB pour le résultat.
- ▶ I : Interrupt Disable
I est activé si le programme a exécuté une instruction de type 'Set Interrupt Disable'. Durant l'activation de ce flag, le processeur ne répondra pas aux demandes d'interruption provenant des périphériques extérieurs, jusqu'à ce que le flag soit désactivé suite à l'exécution d'une instruction de type 'Clear Interrupt Disable'.
- ▶ etc.

Registres

Les registres peuvent se présenter en banc.

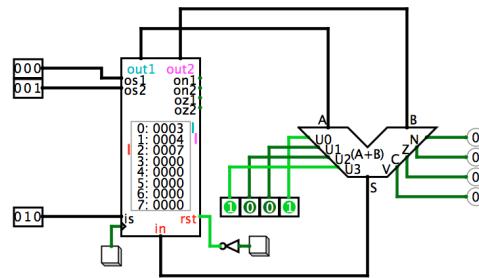
Exemple d'un banc de 8 registres sous Logisim :



- os1 num du registre pour **out1**
- os2 num du registre pour **out2**
- is num du registre à charger avec **in**
- on1 indique si le registre **os1** est négatif
- on2 indique si le registre **os2** est négatif
- on1 indique si le registre **os1** est nul
- on2 indique si le registre **os2** est nul
- clk horloge de chargement de **in**

Illustration

En assemblant l'UAL et le banc de registres, on obtient une calculatrice :



Circuit

Unité de contrôle (décodeur/séquenceur)

- ▶ A besoin de savoir quelle instruction on exécute :
Registre d'instruction **IR**
- ▶ Décode l'instruction à exécuter (OpCode, format)
- ▶ Décompose l'exécution d'une instruction en étapes
- ▶ Donne à chaque étape les signaux :
 - ▶ de sélection de registres
 - ▶ de chargement des registres
 - ▶ du code opération de l'ALU
 - ▶ de connexion de la mémoire ou des registres aux entrées DB/AB
 - ▶ de lecture/écriture en mémoire

Processeur

cycle "Fetch - Decode - Execute"

Préparation : un exemple didactique

traitement à réaliser : addition de deux nombres entiers non-signés

```

debut
  valeur du premier nombre (A) disponible à l'adresse 51
  valeur du second nombre (B) disponible à l'adresse 52
  calculer S ← A + B
  mémoriser le contenu de S à l'adresse 53
fin

```

Préparation : premières instructions

En vue de programmer une addition de deux entiers non signés, on se donne un premier jeu d'instructions à exécuter par notre processeur :

- ▶ chargement du registre A depuis une case mémoire
LDA \$adresse $A \leftarrow M$
charge le registre A avec le contenu de la case mémoire d'adresse **adresse**.
- ▶ écriture du contenu du registre A dans une case mémoire
STA \$adresse $M \leftarrow A$
écrit la valeur du contenu du registre A dans la case mémoire d'adresse **adresse**.
- ▶ addition entre un registre et le contenu d'une case mémoire
ADC \$adresse $A \leftarrow A + M + C$
ajoute au contenu du registre A la valeur du contenu de la case mémoire d'adresse **adresse** plus la valeur du carry C.
- ▶ efface le carry
CLC $C \leftarrow 0$
met à zéro la valeur du carry C.
- ▶ marque la fin du programme
BRK
produit une interruption qui arrête l'exécution du programme.

Préparation : format des instructions

Etude :

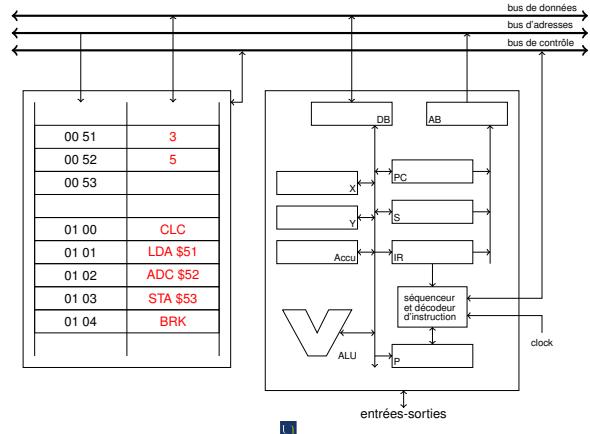
Instruction	mnémonique	opCode	adresse	#bytes	#cycles
		8 bits	8 bit		
LDA \$M	LDA	A5	M	2	3
STA \$M	STA	85	M	2	3
ADC \$M	ADC	65	M	2	3
CLC	CLC	18		1	2
BRK	BRK	00		1	7

Simplification (temporaire) :

abstraction de l'espace occupé en mémoire

→ l'instruction complète (opCode+adresse) représentée sur 8 bits

Préparation : mise en place du programme



Instructions : récapitulatif et instructions proches

Manipulation des flags du registre d'état (P) :

CLC : clear carry
CLI : clear interrupt disable
CLV : clear overflow
SEC : set carry
SEI : set interrupt disable

Mouvements sur les registres A, X et Y :

LDA : load in accumulator
LDX : load in register X
LDY : load in register Y
STA : store in accumulator
STX : store in register X
STY : store in register Y

Autres :

BRK : force Break

Instructions : récapitulatif et instructions proches

Opérations arithmétiques (ALU) :

ADC : add with carry
SBC : subtract with carry

Opérations logiques (ALU) :

AND : and (with accumulator)
ORA : or with accumulator
EOR : exclusive or (with accumulator)

Opérations de décalage (ALU) :

ASL : Shift One Bit Left (memory or accumulator)
LSR : Shift One Bit Right (memory or accumulator)
ROL : Rotate One Bit Left (memory or accumulator)
ROR : Rotate One Bit Right (memory or accumulator)

Processeur (2/2)

(étude détaillée de l'architecture de von Neumann)
 (cycle "Fetch - Decode - Execute")
 modes d'adressage des données
 modes d'adressage pour les branchements
 processeur 6502 : 8 bits de données et 16 bits d'adresses

Processeur

Modes d'adressage des données

Modes d'adressage des données

Vocabulaire général :

- ▶ un opérande :
une donnée qui intervient dans une opération
- ▶ un opérateur :
une opération qui porte sur un ou plusieurs opérandes
- ▶ exemple :
`2 + 5`

Au niveau processeur :

- ▶ un mode d'adressage des données :
une manière d'indiquer comment une valeur est obtenue.
- ▶ exemple :
`LDA $51`

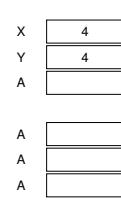
Modes d'adressage des données

Modes non-indexés :

- ▶ **adressage direct (absolu)**
L'opérande est une adresse M *directement* indiquée. C'est à cette adresse que se trouve la donnée : `LDA $51`
- ▶ **adressage immédiat**
L'opérande est une valeur *immédiatement* donnée (sans passer par un registre ou une adresse) : `LDA #$51`
- ▶ **adressage implicite**
L'opérande est *hérité* à l'instruction : `CLC`
- ▶ **adressage par registre**
L'opérande est une valeur contenue dans un *registre* : `TAX`
- ▶ **adressage indirect (pointeur)**
L'opérande est une adresse M indiquée, qui permet d'accéder à une adresse M' en mémoire où se trouve la donnée : `LDA ($51)`

Modes d'adressage des données

00 00	
00 50	B9
00 51	53
00 52	01
00 53	0D
00 54	00
00 55	52
00 56	06
00 57	02
00 58	09
FF FF	



`LDA #$51`
`LDA $51`
`LDA ($51)`
`LDA $51,X`
`LDA ($51,X)`
`LDA ($51),Y`

Simplification (temporaire) : (1) LDA (\$51) supposé disponible et (2) indirection sur une adresse portant sur 1 mot

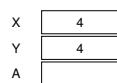
Modes d'adressage des données

Modes indexés :

- ▶ **adressage direct indexé :**
L'opérande est une adresse M *directement* indiquée et incrémentée par la valeur du registre indiqué : `LDA $51,X`
- ▶ **adressage indirect indexé a priori :**
L'opérande est une adresse indiquée et incrémentée par la valeur du registre indiqué, qui permet d'accéder à une adresse en mémoire M' où se trouve la donnée : `LDA ($51,X)`
- ▶ **adressage indirect indexé a posteriori :**
L'opérande est une adresse indiquée, qui permet d'accéder à une adresse en mémoire M' incrémentée par la valeur du registre indiqué, pointant sur l'adresse en mémoire où se trouve la donnée : `LDA ($51),X`

Modes d'adressage des données

00 00	
00 50	B9
00 51	53
00 52	01
00 53	0D
00 54	00
00 55	52
00 56	06
00 57	02
00 58	09
FF FF	



```

LDA #$51      A [ ] 
LDA $51       A [ ] 
LDA ($51)    A [ ] 
LDA $51,X    A [ ] 
LDA ($51,X) A [ ] 
LDA ($51),Y A [ ] 

```

Simplification (temporaire) : (1) LDA (\$51) supposé disponible et (2) indirection sur une adresse portant sur 1 mot

Exemple : parcours d'un vecteur de données (\$53 à \$55)

00 00	
00 50	B9
00 51	53
00 52	01
00 53	0D
00 54	00
00 55	52
00 56	06
00 57	02
00 58	09
FF FF	

```

LDX #$00
LDA $53,X
INX
LDA $53,X
INX
LDA $53,X

```



A []

Instructions (suite) : récapitulatif et instructions proches

Mouvements entre les registres A, X, Y et S :

TAX : transfer accumulator to X
TAY : transfer accumulator to Y
TSX : transfer stack pointer to X
TXA : transfer X to accumulator
TXS : transfer X to stack pointer
TYA : transfer Y to accumulator

Opérations arithmétiques :

INC : increment memory by one
INX : increment X by one
INY : increment Y by one
DEC : decrement memory by one
DEX : decrement X by one
DEY : decrement Y by one

Processeur

Modes d'adressage pour les branchements

Modes d'adressage pour les branchements

Par défaut, les instructions s'exécutent en séquence :

- ▶ quand une instruction est exécutée, on passe à celle qui suit en mémoire.
- ▶ il faut donc toujours connaître l'adresse de la prochaine instruction
- ▶ le processeur mémorise cette adresse dans un registre : le compteur ordinal (PC = *program counter*)
- ▶ lors de l'exécution d'une instruction, le PC est incrémenté pour contenir l'adresse de l'instruction suivante

Instructions qui permettent des ruptures de séquence :

- ▶ notion de branchement
- ▶ renseigner le PC avec l'adresse à laquelle il faut sauter

Modes d'adressage pour les branchements

Adressage absolu

- ▶ usage : brancher n'importe où dans la mémoire (programme)
- ▶ indiquer l'adresse de destination (prochaine instruction à exécuter)
- ▶ l'adresse de destination est donnée dans l'instruction de branchement
- ▶ le contenu du PC est remplacé par l'adresse en question

Remarque :

l'occupation du programme en mémoire est 'figée'

Modes d'adressage pour les branchements

Adressage relatif

- ▶ usage : brancher à proximité de la position actuelle en mémoire
- ▶ indiquer un décalage (offset) positif ou négatif par rapport à l'adresse de la prochaine instruction
- ▶ le décalage vers la destination est donné dans l'instruction de branchement
- ▶ le contenu du PC est remplacé par l'adresse de la destination

Remarque :

l'occupation du programme en mémoire est 'relocalisable' sans besoin d'être modifié

Modes d'adressage pour les branchements

Branchements inconditionnel

JMP : Jump to New Location M
JSR : Jump to New Location M Saving Return Address

avec M=0xHHBB

Modes d'adressage pour les branchements

Branchements conditionnel

- ▶ Tests de condition
- ▶ CMP : compare Memory with Accumulator
- ▶ CPX : compare Memory and X
- ▶ CPY : compare Memory and Y
- ▶ compare les valeurs de M et un registre R (A, X, Y)
- ▶ la comparaison se fait par soustraction : R - M
- ▶ si le résultat est nul, les deux valeurs sont égales
- ▶ si le résultat est négatif, R < M
- ▶ pour utiliser le résultat de la comparaison, il faut mémoriser la valeur des indicateurs N et Z de l'ALU ⇒ registre d'état P.

Modes d'adressage pour les branchements

Instructions de branchements conditionnel :

BEQ : branch on equal (zero set)
BMI : branch on minus (negative set)
BNE : branch on not equal (zero clear)
BPL : branch on plus (negative clear)

BCC : branch on carry clear
BCS : branch on carry set
BVC : branch on overflow clear
BVS : branch on overflow set

Toutes sur 2 bytes ; format : OpCode (1byte) offset (1byte)

Modes d'adressage pour les branchements

NOP : No Operation (1byte)
DEX : Decrement Index X by One (1byte)
BEQ : Branch on Result Zero (2bytes)

Calcul du déplacement :

⋮
PC-4 : FC → 0100 NOP
PC-3 : FD → 0101 DEX z←1 si X=0
0102 BEQ déplacement
PC+0 : 00 → 0104 NOP
PC+1 : 01 → 0105 NOP
⋮

Exemple :

⋮
CMP #\$05
BEQ egal
⋮ code pour le cas où A ≠ \$05
JMP suite
egal ⋮ code pour le cas où A = \$05
suite suite du code
⋮

Équivalent de :

si A = 5 alors ... sinon ... fsi

Instructions (suite) : récapitulatif et instructions proches

Branchements inconditionnel :

JMP : Jump to New Location
JSR : Jump to New Location Saving Return Address
RTI : return from interrupt
RTS : return from subroutine

Test de condition :

CMP : compare Memory with Accumulator
CPX : compare Memory and X
CPY : compare Memory and Y

Branchements conditionnel :

BEQ : branch on equal (zero set)
BMI : branch on minus (negative set)
BNE : branch on not equal (zero clear)
BPL : branch on plus (negative clear)
BCC : branch on carry clear
BCS : branch on carry set
BVC : branch on overflow clear
BVS : branch on overflow set

Autres :

NOP : No Operation

Fonctionnement des ordinateurs



Xavier Gandibleux@Univ-Nantes.fr 36

Ordinogramme

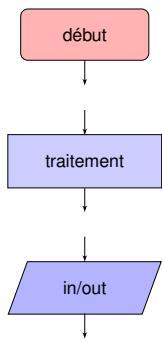
Appelé aussi **organigramme de programmation** : représentation graphique de l'enchaînement des opérations, des décisions ou des fonctions à réaliser à l'aide d'un programme d'ordinateur.

Fonctionnement des ordinateurs



Xavier Gandibleux@Univ-Nantes.fr 37

Ordinogramme

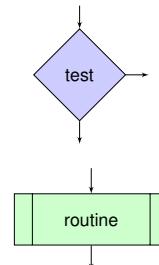


Fonctionnement des ordinateurs



Xavier Gandibleux@Univ-Nantes.fr 38

Ordinogramme



Fonctionnement des ordinateurs



Xavier Gandibleux@Univ-Nantes.fr 39

Processeur

processeur 6502 : 8 bits de données et 16 bits d'adresses



Fonctionnement des ordinateurs



Xavier Gandibleux@Univ-Nantes.fr 40

Présentation

Historique :

- ▶ Constructeurs : MOS Technology, Rockwell, Synertek.
- ▶ Machines : Apple II, Acorn BBC Model B, Atari 800, Commodore 64, Vic 20, PET...

Spécifications de base :

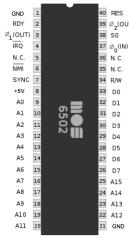
- ▶ Fréquence d'horloge : 1Mhz (6502), 2Mhz (6502-A), 3Mhz (6502-B)
- ▶ Nombre d'instructions : 66
- ▶ Taille d'un mot : 8 bits (Little Endian)
- ▶ Interruptions : logicielle, matérielle et prioritaire non masquable
- ▶ bus d'adresse : 16 bits
- ▶ adressage : de \$0000 à \$FFFF
- ▶ mémoire adressable : 64 Kb
- ▶ stockage adresse 16 bits sur 2 bytes consécutifs, LSByte d'abord
- ▶ modes d'adressage : direct, relatif, immédiat, indexé, implicite, indirect, indirect indexé.
- ▶ Mémoire est vue comme un ensemble de pages de 256 bytes
- ▶ Première page (\$0000 to \$00FF) est appelée 'Page Zero' ; elle peut être accédée en utilisant un mode d'adressage spécial (page zéro) utilisant des instructions plus courtes et plus rapides

Fonctionnement des ordinateurs



Xavier Gandibleux@Univ-Nantes.fr 41

Brochage



DB 0 à DB 7 = Bus de données
AB 0 à AB 15 = Bus d'adresses
R/W = Sélection lecture/écriture
RES = Initialisation
RDY = Synchronisation mémoires lentes
IRQ = Demande d'interruption masquable
NMI = Interruption non masquable
SYNC = Début du cycle recherche d'instruction
SO = Mise à 1 du drapeau de dépassement
GND = Masse
 ϕ_0 = Horloge CPU
 ϕ_1, ϕ_2 = Horloge système
n.c. = non connecté

Clin d'oeil

- ▶ Dans le cyborg tueur de "Terminator" :



- ▶ Dans les "architectures ARM" :

inspiré les architectures ARM ("Acorn RISC Machine" devenu "Advanced Risc Machines", RISC signifiant "Reduced Instruction Set Computing") introduites à partir de 1983 par Acorn Computers (fabricant du BBC model B fonctionnant avec un 6502) et aujourd'hui utilisée dans plusieurs ordinateurs de bureau, systèmes portatifs, systèmes embarqués, téléphonie, etc.

Suite...

Programmation bas niveau