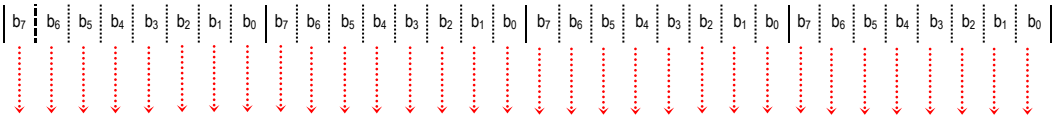
République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

USTHB **Archi**
Faculté d'Electronique et d'Informatique
Département Informatique

Formats des Instructions Machine

INSTRUCTIONS DE TRANSFERT DE DONNÉES



• MOV : *Transferer valeur*

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Reg/Mém ↔ Reg/Mém (Mém↔Mém non autorisé)	1	0	0	0	1	0	d	w	mod		reg			r/m			Dépl (octet bas s'il y'a Dépl relatif)								Dépl (octet haut, s'il y'a Dépl relatif)							
Reg/Mém ↔ Reg/Mém (s'il n'y a pas Dépl Relatif)	1	0	0	0	1	0	d	w	mod		reg			r/m																		
Registre/ Mémoire ← valeur immédiate	1	1	0	0	0	1	1	w	mod	0	0	0	r/m				Donnée (octet bas)								Donnée (octet haut, si w=1)							
Registre ← Immédiate	1	0	1	1	w	reg			Donnée (octet bas)								donnée (octet haut, si w=1)															
Accumulateur ← Mémoire	1	0	1	0	0	0	0	w	Adresse effective (octet bas)								Adr. effective (octet haut, si w=1)															
Mémoire ← Accumulateur	1	0	1	0	0	0	1	w	Adresse effective (octet bas)								Adresse effective (Octet haut)															
Registre segment ← Registre / Mémoire	1	0	0	0	1	1	1	0	mod	0	RSeg		r/m				Déplacement Bas								Déplacement Haut							
Registre / Mémoire ← Registre segment	1	0	0	0	1	1	0	0	mod	0	RSeg		r/m				Déplacement Bas								Déplacement Haut							

• PUSH : *Empiler valeur*

Registre /Mémoire	1	1	1	1	1	1	1	1	mod	1	1	0	r/m				Déplacement Bas								Déplacement Haut							
Registre	0	1	0	1	0	reg																										
Registre segment	0	0	0	RSeg		1	1	0																								

• POP : *Dépiler valeur*

Registre /Mémoire	1	0	0	0	1	1	1	1	mod	0	0	0	r/m																			
Registre	0	1	0	1	1	reg																										
Registre segment	0	0	0	RSeg		1	1	1																								

• XCHG : *Echanger valeur*

Registre / Mémoire ↔ Registre	1	0	0	0	0	1	1	w	mod		reg		r/m																			
Registre ↔ Accumulateur	1	0	0	1	0	reg																										

• IN : *Lecture d'un port d'E/S*

Accumulateur ← Immédiat (8 bits)	1	1	1	0	0	1	0	w	n° du port																							
Accumulateur ← DX	1	1	1	0	1	1	0	w																								

• OUT : *Ecriture dans un port d'E/S*

Immédiat (8 bits) ← Accumulateur	1	1	1	0	0	1	1	w	n° du port																							
DX ← Accumulateur	1	1	1	0	1	1	1	w																								

• XLAT : *Translater*

	1	1	0	1	0	1	1	1

• LEA : *Transférer adresse effective dans registre*

Registre ← Mémoire	1	0	0	0	1	1	0	1	mod		reg			r/m																		

• LDS : *Transférer adresse segment et adresse effective dans DS et registre*

Registre ← Mémoire	1	1	0	0	0	1	0	1	mod		reg			r/m																		

- ***LES :*** *Transférer adresse segment et adresse effective dans ES et registre*

Registre ← Mémoire	1	1	0	0	0	1	0	0	mod	reg	r/m	

- ***LAHF :*** *Transférer le contenu du PSW dans AH*

1	0	0	1	1	1	1	1

- ***SAHF :*** *Transférer le contenu de AH dans PSW*

1	0	0	1	1	1	1	0

- ***PUSHF :*** *Empiler PSW*

1	0	0	1	1	1	0	0

- ***POPF :*** *Depiler PSW*

1	0	0	1	1	1	0	1

INSTRUCTIONS ARITHMÉTIQUES

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
.																																

- ***ADD :*** *Addition*

Reg/Mém ↔ Reg (S'il y'a Dépl Relatif)	0	0	0	0	0	0	d	w	mod		reg		r/m				Dépl (octet bas)								Dépl (octet haut, si w=1)							
Reg / Mém ↔ Reg (s'il pas Dépl. Relatif)	0	0	0	0	0	0	d	w	mod		reg		r/m																			
Registre/ Mémoire ← Valeur immédiate	1	0	0	0	0	0	s	w	mod	0	0	0	r/m				Donnée (octet bas)								donnée (octet haut, si w=1)							
Accumulateur ← Valeur immédiate	0	0	0	0	0	1	0	w	donnée								donnée (si w=1)															

- ***ADC :*** *Addition avec retenue*

Registre / Mémoire ↔ Registre	0	0	0	1	0	0	d	w	mod		reg		r/m																			
Registre/ Mémoire ← Valeur immédiate	1	0	0	0	0	0	s	w	mod	0	1	0	r/m				donnée								donnée (si w=1)							
Accumulateur ← Valeur immédiate	0	0	0	1	0	1	0	w	donnée								donnée (si w=1)															

- ***INC :*** *Incrémentation*

Registre/ Mémoire	1	1	1	1	1	1	1	w	mod	0	0	0	r/m																			
Registre	0	1	0	0	0	reg																										

- ***AAA :*** *Ajustement AscII pour l'addition*

0	0	1	1	0	1	1	1

- ***DAA :*** *Ajustement décimal pour l'addition*

0	0	1	0	0	1	1	1

- ***SUB :*** *Soustraction*

Registre / Mémoire ↔ Registre	0	0	1	0	1	0	d	w	mod		reg		r/m																			
Registre/ Mémoire ← Valeur immédiate	1	0	0	0	0	0	s	w	mod	1	0	1	r/m				donnée								donnée (si w=1)							
Accumulateur ← Valeur immédiate	0	0	1	0	1	1	0	w	donnée								donnée (si w=1)															

- ***SBB :*** *Soustraction avec retenue*

Registre / Mémoire ↔ Registre	0	0	0	1	1	0	d	w	mod		reg		r/m																			
Registre/ Mémoire ← Valeur immédiate	1	0	0	0	0	0	s	w	mod	0	1	1	r/m				donnée								donnée (si w=1)							
Accumulateur ← Valeur immédiate	0	0	0	1	1	1	0	w	donnée								donnée (si w=1)															

- ***DEC :*** *Décrémentation*

Registre/ Mémoire	1	1	1	1	1	1	1	w	mod	0	0	1	r/m																			
Registre	0	1	0	0	1	reg																										

- ***AAS :*** *Ajustement AscII pour la soustraction*

0	0	1	1	1	1	1	1

- ***DAS :*** *Ajustement décimal pour la soustraction*

0	0	1	0	1	1	1	1

- ***NEG :*** *Changement de signe*

Registre/ Mémoire

1	1	1	1	0	1	1	w	mod	0	1	1	r/m

- ***MUL :*** *Multiplication (non signée)*

Registre/ Mémoire

1	1	1	1	0	1	1	w	mod	1	0	0	r/m

- ***IMUL :*** *Multiplication (signée)*

Registre/ Mémoire

1	1	1	1	0	1	1	w	mod	1	0	1	r/m

- ***AAM :*** *Ajustement ascII pour la Multiplication*

1	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0

- ***DIV :*** *Division (non signée)*

Registre/ Mémoire

1	1	1	1	0	1	1	w	mod	1	1	0	r/m

- ***IDIV :*** *Division (signée)*

Registre/ Mémoire

1	1	1	1	0	1	1	w	mod	1	1	1	r/m

- ***AAD :*** *Ajustement ascII pour la Division*

1	1	0	1	0	1	0	1	0	0	0	0	1	0	1	0

- ***CBW :*** *Conversion de Byte en Word*

1	0	0	1	1	0	0	0																				
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

- ***CWD :*** *Conversion de Word en Dword*

1	0	0	1	1	0	0	1

- ***CMP :*** *Comparaison*

Registre / Mémoire ↔ Registre	0	0	1	1	1	0	d	w	mod	Reg		r/m		
Registre/ Mémoire ← Valeur immédiate	1	0	0	0	0	0	s	w	mod	1	1	1	r/m	donnée	donnée (si w=1)
Accumulateur ← Valeur immédiate	0	0	1	1	1	1	0	w	donnée					donnée (si w=1)	

INSTRUCTIONS LOGIQUES

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

- ***AND :*** *ET logique*

Registre / Mémoire ↔ Registre	0	0	1	0	0	0	d	w	mod	reg		r/m		
Registre/ Mémoire ← Valeur immédiate	1	0	0	0	0	0	0	w	mod	1	0	0	r/m	donnée	donnée (si w=1)
Accumulateur ← Valeur immédiate	0	0	1	0	0	1	0	w	donnée					donnée (si w=1)	

- ***OR :*** *Ou logique*

Registre / Mémoire ↔ Registre	0	0	0	0	1	0	d	w	mod	reg		r/m		
Registre/ Mémoire ← Valeur immédiate	1	0	0	0	0	0	0	w	mod	0	0	1	r/m	donnée	donnée (si w=1)
Accumulateur ← Valeur immédiate	0	0	0	0	1	1	0	w	donnée					donnée (si w=1)	

- ***XOR :*** *Ou exclusif logique*

Registre / Mémoire ↔ Registre	0	0	1	1	0	0	d	w	mod	reg		r/m		
Registre/ Mémoire ← Valeur immédiate	1	0	0	0	0	0	0	w	mod	1	1	0	r/m	donnée	donnée (si w=1)
Accumulateur ← Valeur immédiate	0	0	1	1	0	1	0	w	donnée					donnée (si w=1)	

- ***NOT :*** *Non logique*

Registre/ Mémoire

1	1	1	1	0	1	1	w	mod	0	1	0	r/m

- ***TEST :*** *ET logique avec résultat dans PSW*

Registre ← Registre/ Mémoire	1	0	0	0	1	0	0	w	mod	reg		r/m		
Registre/ Mémoire ← Valeur immédiate	1	1	1	1	0	1	1	w	mod	0	0	0	r/m	donnée	donnée (si w=1)
Accumulateur ← Valeur immédiate	1	0	1	0	1	0	0	w	donnée					donnée (si w=1)	

- ***SHL / SAL :*** *Décalage arithmétique logique à gauche*

Registre/ Mémoire ← CL / 1

1	1	0	1	0	0	v	w	mod	1	0	0	r/m

- ***SHR :*** *Décalage logique à droite*

Registre/ Mémoire ← CL / 1

1	1	0	1	0	0	v	w	mod	1	0	1	r/m

- ***SAR :*** *Décalage arithmétique à gauche*

Registre/ Mémoire ← CL / 1

1	1	0	1	0	0	v	w	mod	1	1	1	r/m

- ***ROL :*** *Rotation à gauche*

Registre/ Mémoire ← CL / 1

1	1	0	1	0	0	v	w	mod	0	0	0	r/m

- ***ROR :*** *Rotation à droite*

Registre/ Mémoire ← CL / 1

1	1	0	1	0	0	v	w	mod	0	0	1	r/m

- ***RCL :*** *Rotation à travers la retenue à gauche*

Registre/ Mémoire ← CL / 1

1	1	0	1	0	0	v	w	mod	0	1	0	r/m

- ***RCR :*** *Rotation à travers la retenue à droite*

Registre/ Mémoire ← CL / 1

1	1	0	1	0	0	v	w	mod	0	1	1	r/m

INSTRUCTIONS DE MANIPULATION DES CHAÎNES DE CARACTÈRES

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

- ***REP :*** *Répetition*

1	1	1	1	0	0	1	1

- ***MOVS :*** *Transfert de chaîne de caractères*

Byte/Word

1	0	1	0	0	1	0	w

- ***CMPS :*** *Comparaison de chaînes de caractères*

Byte/Word

1	0	1	0	0	1	1	w

- ***SCAS :*** *Recherche dans chaîne de caractères*

Byte/Word

1	0	1	0	1	1	1	w

- ***LODS :*** *Chargement dans une chaîne de caractères*

Byte/Word

1	0	1	0	1	1	0	w

- ***STOS :*** *Lecture d'une chaîne de caractères*

Byte/Word

1	0	1	0	1	0	1	w

INSTRUCTIONS DE BRANCHEMENT (TRANSFERT DE CONTRÔLE)

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

- ***CALL :*** *Appel de procédure*

Intra-Segment Direct	1	1	1	0	1	0	0	0	Adresse effective (octet bas)				Adresse effective (Octet haut)
Intra-Segment indirect	1	1	1	1	1	1	1	1	mod	0	1	0	r/m
Inter-Segment Direct									Adresse segment (octet bas)				Adresse segment (Octet haut)
Inter-Segment indirect	1	1	1	1	1	1	1	1	mod	0	1	1	r/m

- ***RET :*** *Retour de procédure*

Intra-Segment	1	1	0	0	0	0	1	1	
Intra-Segment avec élimination des paramètres	1	1	0	0	0	0	1	0	donnée (octet bas)	donnée (Octet haut)
Inter-Segment	1	1	0	0	1	0	1	1	
Intre-Segment avec élimination des paramètres	1	1	0	0	1	0	1	0	donnée (octet bas)	donnée (Octet haut)

- ***JMP :*** *Branchement inconditionnel*

Intra-Segment Direct	1	1	1	0	1	0	0	1	Déplacement (octet bas)		Déplacement (Octet haut)
Intra-Segment Direct court	1	1	1	0	1	0	1	1	Déplacement		
Intra-Segment indirect	1	1	1	1	1	1	1	1	mod	1 0 0	r/m
Inter-Segment Direct									Adresse segment (octet bas)		Adresse segment (Octet haut)
Inter-Segment indirect	1	1	1	1	1	1	1	1	mod	1 0 1	r/m

- ***JX :*** *Branchement Conditionnel*

JZ / JE : Branchement si égal ou zéro	0	1	1	1	0	1	0	0	Déplacement
JNZ / JNE : Branchement si non égal ou non zéro	0	1	1	1	0	1	0	1	Déplacement
JL / JNGE : Branchement si inférieur	0	1	1	1	1	1	0	0	Déplacement
JLE / JNG : Branchement si inférieur ou égal	0	1	1	1	1	1	1	0	Déplacement
JNL / JGE : Branchement si supérieur ou égal	0	1	1	1	1	1	0	1	Déplacement
JNLE / JG : Branchement si supérieur	0	1	1	1	1	1	1	1	Déplacement
JB / JNAE : Branchement si inférieur	0	1	1	1	0	0	1	0	Déplacement
JBE / JNA : Branchement si inférieur ou égal	0	1	1	1	0	1	1	0	Déplacement
JNB / JAE : Branchement si supérieur ou égal	0	1	1	1	0	0	1	1	Déplacement
JNBE / JA : Branchement si supérieur	0	1	1	1	0	1	1	1	Déplacement
JP / JPE : Branchement si parité paire	0	1	1	1	1	0	1	0	Déplacement
JNP / JPO : Branchement si Parité impaire	0	1	1	1	1	0	1	1	Déplacement
JS : Branchement si signée	0	1	1	1	1	0	0	0	Déplacement
JNS : Branchement si non signée	0	1	1	1	1	0	0	1	Déplacement
JO : Branchement si overflow	0	1	1	1	0	0	0	0	Déplacement
JNO : Branchement si non overflow	0	1	1	1	0	0	0	1	Déplacement
JCXZ : Branchement si CX est égal à zéro	1	1	1	0	0	0	1	1	Déplacement
LOOP : Boucle	1	1	1	0	0	0	1	0	Déplacement
LOOPZ/LOOPE : Boucle tant que égal ou zéro	1	1	1	0	0	0	0	1	Déplacement
LOOPNE/LOOPNZ : Boucle tant que non égal ou non zéro	1	1	1	0	0	0	0	0	Déplacement

- ***INT :*** *Appel d'interruption*

Numéro de l'interruption	1	1	0	0	1	1	0	1	Numèro
INT 03	1	1	0	0	1	1	0	0	
INT0	1	1	0	0	1	1	1	0	

- ***IRET*** *: Retour d'interruption*

	1	1	0	0	1	1	1	1

INSTRUCTIONS DE CONTRÔLE DU REGISTRE D'ETAT DU MICROPROCESSEUR

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

- ***CLC*** *: Remise à 0 du bit CF*

	1	1	1	1	1	0	0	0

- ***STC*** *: Mise à 1 du bit CF*

	1	1	1	1	1	0	0	1

- ***CMC*** *: Inversion du bit CF*

	1	1	1	1	0	1	0	1

- ***CLD*** *: Remise à 0 du bit DF*

	1	1	1	1	1	1	0	0

- ***STD*** *: Mise à 1 du bit DF*

	1	1	1	1	1	1	0	1

- ***CLI*** *: Remise à 0 du bit IF*

1	1	1	1	1	0	1	0

- ***STI*** *: Mise à 1 du bit IF*

1	1	1	1	1	0	1	1

- ***HLT*** *: Arrêt attente interruption*

1	1	1	1	0	1	0	0

- ***WAIT*** *: Attente du signal test du Coprocesseur*

1	0	0	1	1	0	1	1

- ***ESC*** *: Code échappement qui indique que l'instruction suivante est destiné au coprocesseur*

1	1	0	1	1	x	x	x	mod	x	x	x	r/m

- ***LOCK*** *: Verrouillage du bus système*

1	1	1	1	0	0	0	0

MODES D'ADRESSAGE (REGISTRES SEGMENTS PAR DÉFAUT SONT DS ET SS) ET CODES DES REGISTRES

MOD / R/M	00	01	10	11	
				W = 0	W = 1
000	[BX+SI] DS	Dépl8[BX+SI] DS	Dépl16[BX+SI] DS	AL	AX
001	[BX+DI] DS	Dépl8[BX+DI] DS	Dépl16[BX+DI] DS	CL	CX
010	[BP+SI] SS	Dépl8[BP+SI] SS	Dépl16[BP+SI] SS	DL	DX
011	[BP+DI] SS	Dépl8[BP+DI] SS	Dépl16[BP+DI] SS	BL	BX
100	[SI] DS	Dépl8[SI] DS	Dépl16[SI] DS	AH	SP
101	[DI] DS	Dépl8[DI] DS	Dépl16[DI] DS	CH	BP
110	Dépl16 DS	Dépl8[BP] SS	Dépl16[BP] SS	DH	SI
111	[BX] DS	Dépl8[BX] DS	Dépl16[BX] DS	BH	DI

Registres	Codes
AL et AX	000
CL et CX	001
DL et DX	010
BL et BX	011
AH et SP	100
CH et BP	101
DH et SI	110
BH et DI	111
ES	00
CS	01
SS	10
DS	11

FORCEMENT DES REGISTRES SEGMENTS PAR DÉFAUTS PAR D'AUTRES REGISTRES SEGMENTS

Registre Pointeur ou Indexe	Registre Segment Par Défaut	Qui peut être forcé par
IP	CS	aucun
SP	SS	aucun
BP	SS	DS, ES, ou CS
SI ou DI (pas pour manipuler chaîne de caractères)	DS	ES, SS ou CS
SI	DS	ES, SS ou CS
DI	ES	aucun

Remarques :

1. Dans le cas de forcement d'un segment par défaut par un autre, on doit ajouter le préfixe (1 octet) suivant à l'instruction machine :

Préfixe à ajouter au code machine	Cycles d'horloge supplémentaires
0 0 1 RSeg 1 1 0	2

2. Les valeurs du bit ***D*** sont :

$$D = \begin{cases} 1 & \text{Si argument destinataire est un registre} \\ 0 & \text{Si argument source est un registre} \end{cases}$$

3. Les valeurs du bit ***W*** sont :

$$W = \begin{cases} 1 & \text{Si la donnée manipulée est sur 16 bits} \\ 0 & \text{Si la donnée est sur 8 bits} \end{cases}$$

4. Les valeurs du bit ***S*** sont :

$$S = \begin{cases} 1 & \text{Si l'opération est signée} \\ 0 & \text{Sinon} \end{cases}$$

5. Les valeurs du bit ***V*** sont :

$$V = \begin{cases} 1 & \text{Si le reg. CL est utilisé dans décalage/rotation} \\ 0 & \text{Sinon} \end{cases}$$