

ENSEIRB

**DEPARTEMENT ELECTRONIQUE  
E2  
- STAGE 68000 -  
Cours 3 : Instructions de branchement**

Patrice KADIONIK

email : kadionik@enseirb.fr  
http : www.enseirb.fr/~kadionik

ENSEIRB/E2

Stage 68000



© P. Kadionik - Reproduction soumise à l'accord préalable de l'auteur - 1 / 44 -

---

---

---

---

---

---

---

---

---

---

SOMMAIRE

1. OBJECTIFS
2. INSTRUCTIONS DE BRANCHEMENT INCONDITIONNEL
3. INSTRUCTIONS DE BRANCHEMENT CONDITIONNEL
4. INSTRUCTION DE TESTS AUX LIMITES
5. PRESENTATION DU LANGAGE ASSEMBLEUR
6. CHAINE D 'ASSEMBLAGE ET D 'EDITION DE LIENS

ENSEIRB/E2

Stage 68000



© P. Kadionik - Reproduction soumise à l'accord préalable de l'auteur - 2 / 44 -

---

---

---

---

---

---

---

---

---

---

1. OBJECTIFS :

- Étude des instructions de branchement :
  - Inconditionnel : saut JMP et branchement BRA.
  - Conditionnel : Bcc, DBcc, CHK soumis au test de flag(s) du registre CCR (*Code Condition Register*).
- Présentation de la syntaxe du langage assembleur :
  - Structuration.
  - Directives d 'assemblage.
- Présentation de la chaîne d 'assemblage et d 'édition de liens.

ENSEIRB/E2

Stage 68000



© P. Kadionik - Reproduction soumise à l'accord préalable de l'auteur - 3 / 44 -

---

---

---

---

---

---

---

---

---

---

## 2. INSTRUCTIONS DE BRANCHEMENT INCONDITIONNEL :

- Les instructions sont rangées de façon séquentielle en mémoire et l'exécution d'un programme se fait de façon linéaire.
- Un programme contient des instructions de déroutement et fait appel à des sous programmes pour dérouter cette exécution séquentielle.
- Dans sa plus simple expression, un programme englobe une boucle infinie...

ENSEIRB/E2

Stage 68000



© P.Kadiouik - Reproduction soumise à l'accord préalable de l'auteur - 4 / 44 -

---

---

---

---

---

---

---

---

- On a pour cela :
  - des instructions de déroutement : JMP, BRA, Bcc, DBcc.
  - des instructions d'appel de sous programme : JSR, BSR.
- On distingue les instructions qui utilisent un adressage absolu : **JMP** et JSR qui permettent de transférer le contrôle du programme n'importe où dans les 16 Mo d'espace d'adressage du 68000.
- On distingue les instructions qui utilisent un adressage relatif par rapport à la valeur courante du PC : **BRA** et BSR. Ce déplacement peut être court et codé sur un octet (-128 à +127) ou long codé sur 2 octets (-32768 à + 32767).

ENSEIRB/E2

Stage 68000



© P.Kadiouik - Reproduction soumise à l'accord préalable de l'auteur - 5 / 44 -

---

---

---

---

---

---

---

---

- Saut :  
JMP <EA>  
PC <- (<EA>)

Taille des opérandes : non défini  
Modes d'adressage : tous sauf immédiat  
Flags du CCR : non affectés

ex :  
JMP (A0)  
JMP \$3000  
JMP SCE(PC)

ENSEIRB/E2

Stage 68000



© P.Kadiouik - Reproduction soumise à l'accord préalable de l'auteur - 6 / 44 -

---

---

---

---

---

---

---

---

• Branchement :

BRA <étiquette>

PC ← PC + d d en complément à 2

Taille des opérandes : S (d sur 1 octet) , L (d sur 2 octets)

Modes d'adressage : relatif par rapport à la valeur courante du PC

Flags du CCR : non affectés

- PC pointe toujours sur l'instruction suivante à exécuter.
- Le déplacement étant limité, on n'accède pas à toute la mémoire par rapport à la valeur courante du PC.

ENSEIRB/E2

Stage 68000




---

---

---

---

---

---

---

---

---

---

• d8 :

BRA.S

-128 ≤ d8 ≤ +127

• d16 :

BRA.L

-32768 ≤ d16 ≤ +32767

• OPCODE d8 : |01100000| d8 | (2 octets)

• OPCODE d16 : |01100000|00000000|  
| d16 | (4 octets)

ENSEIRB/E2

Stage 68000




---

---

---

---

---

---

---

---

---

---

• Exemple : assembler à la main le code suivant :

\$1000 ...

...

\$1030 BRA.S ...

\$1032 ...



• Calcul du déplacement d8 pour se rebrancher à l'adresse \$1000.

• PC vaut \$1032 à l'exécution de BRA.S



ENSEIRB/E2

Stage 68000




---

---

---

---

---

---

---

---

---

---

•  $D8 = \$1000 - \$1032 = -\$32$

•  $-\$32 = \text{Compl2}(\$32)$  :

00110010

11001101 (Compl1)

+1

11001110 soit SCE

• Assemblage à la main donne : \$60 SCE pour BRA.S

• Si l'on avait choisi BRA.L, on aurait « gaspillé » 2 octets en plus (PC=\$1034 alors).

ENSEIRB/E2

Stage 68000



---

---

---

---

---

---

---

---

• L'étiquette correspond à une adresse symbolique.

• Le programme assembleur va remplacer cette étiquette par le déplacement correspondant.

• Exemple :

\$1000 DEBUT ....

....

\$1030 BRA.S DEBUT

\$1032 ....

ENSEIRB/E2

Stage 68000



---

---

---

---

---

---

---

---

• Le déplacement calculé reste valable quelle que soit la position du programme en mémoire car c'est un déplacement relatif.

• On a un programme qui a une qualité essentielle : il est relogeable (n'importe où en mémoire)

• ON PREFERERA DONC BRA A JMP (SAUF EXCEPTION) et DE MEME BSR A JSR !

ENSEIRB/E2

Stage 68000



---

---

---

---

---

---

---

---

### 3. INSTRUCTIONS DE BRANCHEMENT CONDITIONNEL :

- Les instructions de branchement conditionnel sont soumises au test d'un ou plusieurs flags du registre de Code Condition CCR.
- On distingue les instructions :
  - de branchement conditionnel Bcc.
  - de test, décrémentation et branchement : DBcc.
  - de mise à \$FF ou à \$00 d'un octet mémoire : Scc.
- On ne donne pas tous les mnémoniques de ces instructions mais leur forme générale :
  - Bcc, DBcc... où le suffixe cc (Code Condition) indique la ou les conditions testées.

ENSEIRB/E2

Stage 68000




---

---

---

---

---

---

---

---

---

---

- 16 conditions cc sont testables :

SUFFIXE cc	CONDITION	VRAI SI
EQ	Equal to	Z=1
NE	Not Equal	Z=0
MI	Minus	
PL	Plus	
GT	Great Than	
LT	Less Than	
GE	Great than or Equal	
LE	Less than or Equal	

ENSEIRB/E2

Stage 68000




---

---

---

---

---

---

---

---

---

---

SUFFIXE cc	CONDITION	VRAI SI
HI	Higher than	
LS	Low than or Same as	
CS	Carry Set	C=1
CC	Carry Clear	C=0
VS	oVerflow Set	V=1
VC	oVerflow Clear	V=0
T	always True	- (= BRA)
F	always False	

ENSEIRB/E2

Stage 68000




---

---

---

---

---

---

---

---

---

---

- DBcc et Sec utilisent les 16 conditions
- Bcc n'utilise pas T et F (14 conditions).
- On distingue le test :
  - de conditions simples : un seul flag est testé.
  - de conditions multiples : plusieurs flags sont testés.




---

---

---

---

---

---

---

---

---

---

- Le test de conditions multiples fait apparaître une distinction entre nombre Non Signé et nombre Signé :

	Nombre NS	Nombre S
>	HI	GT
>=	CC	GE
<	CS	LT
<=	LS	LE
	C,Z	V,Z,N




---

---

---

---

---

---

---

---

---

---

- Le flag N intervient toujours dans les équations logiques de ces instructions de branchement lorsque l'on manipule des nombres Signés.
- C'est à nous de savoir le type des entiers manipulés et donc utiliser la bonne instruction de branchement conditionnel en conséquence (ET NON LE PROCESSEUR !).




---

---

---

---

---

---

---

---

---

---

- Branchement conditionnel :  
   Bcc <étiquette>  
     Si (condition vraie)  
       PC <- PC + d

Taille des opérandes : S (d sur 1 octet) , L (d sur 2 octets)  
 Modes d 'adressage : d(PC)  
 Flags du CCR : non affectés

- Forte analogie avec l 'instruction BRA.




---

---

---

---

---

---

---

---

- Les instruction Bcc sont utilisées de paire avec les instructions de comparaison suivant le schéma :

- 1. Comparaison pour l 'affectation des bits du registre CCR.
- 2. Branchement conditionnel suivant la ou les valeurs des bits du registre CCR.

- Exemple : test d 'un registre d 'un périphérique

```
MOVE.B  $F00000,D1
CMPI.B  #$80,D1
BEQ     TRAITEMENT_PERI
```




---

---

---

---

---

---

---

---

- Test condition, décrémentation et branchement conditionnel :  
   DBcc Dn,<étiquette>  
     Si (condition fausse)  
       PC <- PC + d

Taille des opérandes : L (d sur 2 octets)  
 Modes d 'adressage : d(PC)  
 Flags du CCR : non affectés

- Forte analogie avec l 'instruction BRA.




---

---

---

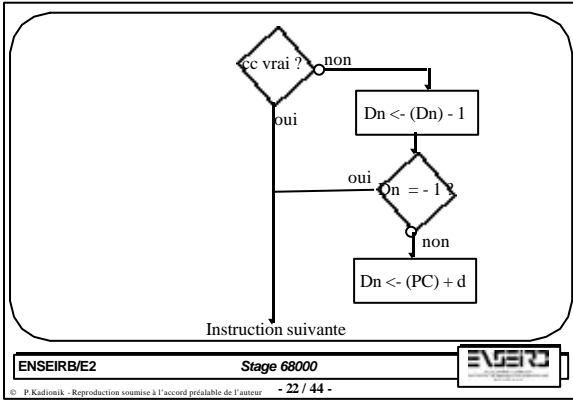
---

---

---

---

---




---

---

---

---

---

---

---


---

- Si Dn est initialisé à n, on parcourt (n+1) fois la boucle si le test cc est toujours faux.
- Exemple :
 

```

      MOVE.B #4,D0
      BCL ...
      DBF D0,BCL
      
```

 on parcourt 5 fois la boucle BCL !
- DBcc est utilisé pour écrire des boucles (de temporisation, de scrutation...) : on réalise à la fois la comparaison et le test...

ENSEIRB/E2 Stage 68000 

© P. Kadionik - Reproduction soumise à l'accord préalable de l'auteur - 23 / 44 -

---

---

---

---

---


---

---

---

Comparaison Bcc et DBcc :

- Bcc : branchement si cc vraie vs DBcc : branchement si cc fausse.
- DBcc : 2 façons de quitter la boucle : cc vraie ou Dn = -1.
- Bcc déplacement d8 ou d16 vs DBcc déplacement toujours d16.

ENSEIRB/E2 Stage 68000 

© P. Kadionik - Reproduction soumise à l'accord préalable de l'auteur - 24 / 44 -

---

---

---

---

---

---

---

---

- positionnement conditionnel d'un octet mémoire :

Scc <EA>

Si (condition vraie)

<EA> <- \$FF

sinon

<EA> <- \$00

Taille des opérandes : B

Modes d'adressage :

Flags du CCR : non affectés

- Positionnement d'indicateurs octet et non bit.

ENSEIRB/E2

Stage 68000




---

---

---

---

---

---

---

---

---

---

#### 4. INSTRUCTION DE TESTS AUX LIMITES :

CHK <EA>,Dn

Taille des opérandes : W

Modes d'adressage : tous

Flags du CCR : affectés

- Test de la valeur d'un registre de donnée par rapport à un intervalle borné.

ENSEIRB/E2

Stage 68000




---

---

---

---

---

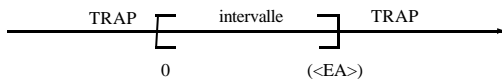
---

---

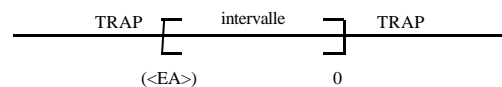
---

---

---



ou



ENSEIRB/E2

Stage 68000




---

---

---

---

---

---

---

---

---

---

- Si l'on se trouve dans l'intervalle, on passe à l'instruction suivante.
- Si l'on est hors intervalle, il y a génération d'une exception logicielle TRAP (piège) possédant le numéro de vecteur 6 dans la table des vecteurs (4 octets pour préciser l'adresse du point d'entrée de la routine d'exception exécutée dans ce cas).

• Exemple :

```

MOVE.L    #5,D0
CHK      #5,D0      pas de trap
....
MOVE.L    #5,D0
CHK      #4,D0      trap

```




---

---

---

---

---

---

---

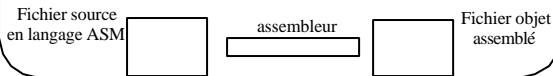
---

---

---

5. PRESENTATION DU LANGAGE ASSEMBLEUR :

- Le microprocesseur 68K ne comprend et n'exécute qu'une suite de valeurs binaires sous forme d'octets rangés en mémoire.
- On utilise un langage compréhensible par l'humain (fichier source ASCII) et on utilise un logiciel qui va générer cette suite d'octets :
  - le fichier source ASCII correspond au programme assembleur ASM.
  - Le logiciel est le programme assembleur ou assembleur.




---

---

---

---

---

---

---

---

---

---

INSTRUCTIONS EN LANGAGE ASM :

- Un programme source est une suite logique d'instructions pour réaliser une tâche particulière.
- Une ligne source est soit :
  - Une instruction en langage ASM ou langage machine.
  - Un commentaire.
  - Une directive d'assemblage.




---

---

---

---

---

---

---

---

---

---

LIGNE D'INSTRUCTION :

[ETIQUETTE] MNEMONIQUE [OPERANDES] [COMMENTAIRE]  
caractère(s) ESPACE ou tabulation(s)

- Champ ETIQUETTE : n'importe quelle instruction peut avoir une étiquette. C'est une chaîne de caractère de 1 à 30 caractères alphanumériques dont le premier caractère est alphabétique. Seules, les 8 premières lettres de l'étiquette apparaissent dans la table des symboles.
- Champ MNEMONIQUE : acronyme de l'instruction ASM. Un mnémonique peut avoir 0, 1 ou 2 opérandes. Le mnémonique possède un suffixe .B, .W (par défaut si omis) ou .L. IL FAUT TOUJOURS LE PRECISER SOUS PEINE D'EFFETS DE BORD !



---

---

---

---

---

---

---

---

---

---

LIGNE D'INSTRUCTION :

[ETIQUETTE] MNEMONIQUE [OPERANDES] [COMMENTAIRE]  
caractère(s) ESPACE ou tabulation(s)

- Champ OPERANDES : paramètre(s) du mnémonique. 0, 1 ou 2 opérandes. S'il y a 2 opérandes, ils sont séparés par une virgule.
- Champ COMMENTAIRE : optionnel mais à mettre absolument ! Il faut commenter ses programmes sources. Sera-t-on capable de relire son programme quelques semaines plus tard ou quelqu'un d'autre ?



---

---

---

---

---

---

---

---

---

---

LIGNE DE COMMENTAIRE :

- A tout moment, on peut insérer une ligne de commentaire. Elle commence par le caractère \*



---

---

---

---

---

---

---

---

---

---



DIRECTIVES D'AFFECTION MEMOIRE :

- Définition de données initialisées (constantes) : DC (*Define Constant*)
- Définition d'une zone mémoire contigue non initialisée : DS (*Define Storage*)

TABLE	DC.W	10,5,7,2
CHAINE	DC.B	'ceci est une chaine ascii '
TEMPO	DS.B	10           réservation de 10 octets

- On utilise les suffixes .B, .W et .L.



---

---

---

---

---

---

---

---

- Si l'on regarde le type des octets en mémoire, on distingue :

- les octets correspondants au code exécutable rangés une zone ou section appelée section *text*.
- Les octets correspondants aux données initialisées (variables initialisées, DC) rangés dans une section *data* (vars avec le compilateur croisé C 68000).
- Les octets correspondants aux données non initialisées (variables non initialisées, DS) rangés dans une section *bss* (*Bloc Starting by Symbol*) (zerovars avec le compilateur croisé C 68000).



---

---

---

---

---

---

---

---

AUTRES DIRECTIVES :

- Assemblage conditionnel.
- Définition de macro-instruction.
- Voir poly (ou [www.enseirb.fr/~kadionik](http://www.enseirb.fr/~kadionik)).



---

---

---

---

---

---

---

---

6. CHAÎNE D'ASSEMBLAGE ET D'ÉDITION DE LIENS :

- Dans la chaîne de développement, on distingue différentes étapes :
1. Édition d'un fichier source en langage ASM à l'aide d'un éditeur de texte (emacs, vi...) : filename.s
  2. Assemblage du fichier source avec l'assembleur et génération du fichier objet filename.o
  3. Édition de liens des différents fichiers objets (fichiers bibliothèque) pour générer un fichier téléchargeable en mémoire : filename.o + filename2.o + filename3.o + bibliotheque.o = final.x



---

---

---

---

---

---

---

---

4. Téléchargement du fichier final.x dans la mémoire (RAM) du kit 68000 (ou cible).

5. Exécution, debugging du programme.

- L'édition de liens permet de fixer les adresses de base des différentes sections (text, data, bss). On résout aussi les références croisées. Un fichier final.map donne le mapping mémoire occupé par le programme et ses différentes sections. On y retrouve aussi incluse la table de symboles donnant l'association (symbole, adresse).



---

---

---

---

---

---

---

---

- Le fichier final.x est un fichier ASCII (éditable) respectant un format particulier : fichier S Record (format Motorola) :

An S-record file consists of a sequence of specially formatted ASCII character strings. An S-record will be less than or equal to 78 bytes in length. The order of S-records within a file is of no significance and no particular order may be assumed.

The general format of an S-record follow:

```
+-----//-----//-----+  
| type | count | address | data | checksum |  
+-----//-----//-----+
```



---

---

---

---

---

---

---

---

- **type** A char[2] field. These characters describe the type of record (S0, S1, S2, S3, S5, S7, S8, or S9).
- **count** A char[2] field. These characters when paired and interpreted as a hexadecimal value, display the count of remaining character pairs in the record.
- **address** A char [4,6, or 8] field. These characters grouped and interpreted as a hexadecimal value, display the address at which the data field is to be loaded into memory. The length of the field depends on the number of bytes necessary to hold the address. A 2-byte address uses 4 characters, a 3-byte address uses 6 characters, and a 4-byte address uses 8 characters.

ENSEIRB/E2

Stage 68000




---

---

---

---

---

---

---

---

---

---

- **data** A char [0-64] field. These characters when paired and interpreted as hexadecimal values represent the memory loadable data or descriptive information.
- **checksum** A char[2] field. These characters when paired and interpreted as a hexadecimal value display the least significant byte of the ones complement of the sum of the byte values represented by the pairs of characters making up the count, the address, and the data fields.

S00600004844521B

S1130000285F245F2212226A000424290008237C2A

.... (19 octets add (2)+data (16)+checksum(1)

S9030000FC

ENSEIRB/E2

Stage 68000




---

---

---

---

---

---

---

---

---

---