# Exercice 2

Analyser un programme et comprendre son fonctionnement.

## 1. Prérequis

Connaitre les fonctionnalités du logiciel MOTO6809.
Connaitre les instructions du MOTO6809 (voir annexe, les instructions utiles sont spécifiées)

## 2. Durée : 1h

Questions d'introduction à l'exercice : **15 mn**

Remplissage du tableau : **45 mn**

- 1ere itération (17 mn)
- 2e à 5e itération (7 x 4mn = 28mn)

## 3. Consignes

- Exécuter le programme pas à pas
- Remplir le tableau des valeurs mises à jour
- Commenter chaque ligne
- Déterminer la fonctionnalité du programme

## 4. Activité

- Analyser le programme et répondre aux questions
- Lancer l'émulateur MOTO6809
- Charger le programme exemple.as9

## 5. Questions d'introduction au programme

### a. Langage des instructions


Que signifie
ld ? : load, mettre la valeur dans le registre …
ldb ? : load dans b, mettre la valeur dans b
ldx ? : load dans x, mettre la valeur dans x
clr ? : clear, effacer le registre …
clra ? : clear a, effacer (mettre 0) dans le registre a
st ? : store, mettre dans la mémoire la valeur …
sta $0000 ? : store a à l'adresse mémoire $0000
cmp ? : compare, fait une comparaison et modifie les flags concernés
cmpa $0000 : compare la valeur du registre a avec la valeur de l'adresse mémoire $0000. Modifier les flags concernés
bcs ? : branch if carry=1, faire un saut si le bit de retenu est à 1

### b. Instruction du programme

A partir de quelle adresse sont situées les données ? $0001
Combien de données y a-t-il ? 6
Quelle est la base des données ? base 16, hexadécimale
Dans quel registre sont traitées les données une à une, à chaque itération ? : le registre a (lda ,x+)
Quel est le registre qui contient l'adresse de la prochaine donnée à traiter ? : le registre x (lda ,x+)

## 6. Tableau des valeurs à renseigner


A chaque tour de boucle, renseigner les valeurs en précisant le numéro d'itération concernée :
Exemple : *itération : valeur*

| instruction | Mémoire $0000 | PC | Nombre | Debut Nbr | A | B | C | X | Commentaires |
|---|---|---|---|---|---|---|---|---|---|
| Valeurs de début => | **0 :** 05<br>**1 :** 67<br>**2 :** F5<br>. | **1 :** FC00 | | | **1 :** 00 | **1 :** 00 | **1 :** 0 | **1 :** 0000 | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | .<br>**5 :** 72 | | | | | | | | |
| *liste des données | | | | | | | | | |
| | | | | | | | | | Initialiser la mémoire ram à l'adresse $00001 à la valeur $05 |
| ;$0001 db $05 | | | | | | | | | |
| ;$0002 db $67 | | | | | | | | | |
| ;$0003 db $F5 | | | | | | | | | |
| ;$0004 db $79 | | | | | | | | | |
| ;$0005 db $e3 | | | | | | | | | |
| ;$0006 db $72 | | | | | | | | | |
| | | | | | | | | | |
| Nombre EQU $0001 | | | **1 :** 05 | | | | | | Lit la mémoire à l'adresse 0001 et met la valeur dans Nombre |
| DebutNbr EQU $0002 | | | | **1 :** 02 | | | | | Lit la mémoire à l'adresse 0002 et met la valeur dans DebutNbr |
| | | | | | | | | | |
| **prgpal:** | | | | | | | | | Etiquette pour les sauts |
| ldb Nombre | | | | | | **1 :** 05 | | | Mettre Nombre dans le registre b |
| clr $0000 | | **1 :** FC03 | | | | | | | Reset de la mémoire à l'adresse 0000 |
| clra | | **1 :** FC06 | | | **1 :** 00 | | | | Reset du registre A |
| ldx #DebutNbr | | **1 :** FC07 | | | | | | **1 :** 0002 | Charge dans le registre X la valeur de la variable DebutNbr |
| | | | | | | | | | |
| **maxm:** | | | | | | | | | |
| lda ,x+ | | **1 :** FC0A<br>**2 :** FC0A | | | **1 :** 67<br>**2 :** F5 | | | **1 :** 0003<br>**2 :** 0004 | Met dans le registre A la valeur à l'adresse mémoire représentée par X puis passer à la ligne suivante de la mémoire |
| cmpa $0000 | | **1 :** FC0C<br>**2 :** FC0C | | | | | **1 :** 0<br>**2 :** 1<br>**3 :** 0<br>.<br>**5 :** 0 | | Fait une comparaison de A avec la valeur de la mémoire en adresse $0000. Si A est supérieure alors 'C'arry = 1 |
| bcs nochg | | **1 :** FC0F | | | | | | | Saut vers nochg si C = 0 |
| sta $0000 | **1 :** 67 | **1 :** FC11 | | | | | | | Mettre le contenu de A dans l'adresse $0000 |
| | | | | | | | | | |
| **nochg:** | | | | | | | | | |
| decb | | **1 :** FC14<br>**2 :** FC14 | | | | **1 :** 04<br>**2 :** 03<br>.<br>.<br>**5 :** 00 | | | Décrémente B |
| bne maxm | | **1 :** FC15<br>**2 :** FC15 | | | | | **1 :** 0<br>**2 :** 0<br>.<br>.<br>**5 :** 1 | | Aller à maxm si la B non égal à 0 |
| end | | | | | | | | | |
| | | | | | | | | | |
| **nochg:** | | | | | | | | | |
| | | | | | | | | | |

## 7. Explications

A chaque tour, une valeur est lue à partir de valeur de l'adresse stockée dans le registre x.

La valeur du nombre d'éléments restant à lire est décrémenté et stocké dans le registre b. Ce nombre est comparé à 0 pour sortir ou non de la boucle.

Le programme compare la valeur stockée à l'adresse de la mémoire $0000 et la valeur lue. Si cette dernière est supérieure à celle mémorisée alors elle la remplace.

On obtient la valeur maximale de la liste.

# Annexe

# INSTRUCTIONS DU 6809

6809 Instruction Set

| Instruction | Mnemonic | Addressing Mode | | | | | | | | | | | | | | | Description | CC bit | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Immediate | | | Direct | | | Indexed | | | Extended | | | Inherent | | | | 5 | 3 | 2 | 1 | 0 |
| | | Op | ~ | # | Op | ~ | # | Op | ~ | # | Op | ~ | # | Op | ~ | # | | H | N | Z | V | C |
| ABX | | | | | | | | | | | | | | 3A | 3 | 1 | X = B+X (Unsigned) | | | | | |
| ADC | ADCA | 89 | 2 | 2 | 99 | 4 | 2 | A9 | 4+ | 2+ | B9 | 5 | 3 | | | | A = A+M+C | + + | + + | + + | + + | + + |
| | ADCB | C9 | 2 | 2 | D9 | 4 | 2 | E9 | 4+ | 2+ | F9 | 5 | 3 | | | | B = B+M+C | + | + | + | + | + |
| ADD | ADDA | 8B | 2 | 2 | 9B | 4 | 2 | AB | 4+ | 2+ | BB | 5 | 3 | | | | A = A+M | + | + | + | + | + |
| | ADDB | CB | 2 | 2 | DB | 4 | 2 | EB | 4+ | 2+ | FB | 5 | 3 | | | | B = B+M | + | + | + | + | + |
| | ADDD | C3 | 4 | 3 | D3 | 6 | 2 | E3 | 6+ | 2+ | F3 | 7 | 3 | | | | D = D+M:M+1 | | + | + | + | + |
| AND | ANDA | 84 | 2 | 2 | 94 | 4 | 2 | A4 | 4+ | 2+ | B4 | 5 | 3 | | | | A = A && M | | + | + | 0 | |
| | ANDB | C4 | 2 | 2 | D4 | 4 | 2 | E4 | 4+ | 2+ | F4 | 5 | 3 | | | | B = B && M | | + | + | 0 | |
| | ANDCC | 1C | 3 | 2 | | | | | | | | | | | | | C = CC && IMM | ? | ? | ? | ? | ? |
| ASL | ASLA | | | | | | | | | | | | | 48 | 2 | 1 | Arithmetic shift left | 8 | + | + | + | + |
| | ASLB | | | | | | | | | | | | | 58 | 2 | 1 | | 8 | + | + | + | + |
| | ASL | | | | 08 | 6 | 2 | 68 | 6+ | 2+ | 78 | 7 | 3 | | | | | 8 | + | + | + | + |
| ASR | ASRA | | | | | | | | | | | | | 47 | 2 | 1 | Arithmetic shift right | 8 | + | + | | + |
| | ASRB | | | | | | | | | | | | | 57 | 2 | 1 | | 8 | + | + | | + |
| | ASR | | | | 07 | 6 | 2 | 67 | 6+ | 2+ | 77 | 7 | 3 | | | | | 8 | + | + | | + |
| BIT | BITA | 85 | 2 | 2 | 95 | 4 | 2 | A5 | 4+ | 2+ | B5 | 5 | 3 | | | | Bit Test A (M&&A) | | + | + | 0 | |
| | BITB | C5 | 2 | 2 | D5 | 4 | 2 | E5 | 4+ | 2+ | F5 | 5 | 3 | | | | Bit Test B (M&&B) | | + | + | 0 | |
| CLR | CLRA | | | | | | | | | | | | | 4F | 2 | 1 | A = 0 | | 0 | 1 | 0 | 0 |
| | CLRB | | | | | | | | | | | | | 5F | 2 | 1 | B = 0 | | 0 | 1 | 0 | 0 |
| | CLR | | | | 0F | 6 | 2 | 6F | 6+ | 2+ | 7F | 7 | 3 | | | | M = 0 | | 0 | 1 | 0 | 0 |
| CMP | CMPA | 81 | 2 | 2 | 91 | 4 | 2 | A1 | 4+ | 2+ | B1 | 5 | 3 | | | | Compare M from A | 8 | + | + | + | + |
| | CMPB | C1 | 2 | 2 | D1 | 4 | 2 | E1 | 4+ | 2+ | F1 | 5 | 3 | | | | Compare M from B | 8 | + | + | + | + |
| | CMPD | 10 83 | 5 | 4 | 10 93 | 7 | 3 | 10 A3 | 7+ | 3+ | 10 B3 | 8 | 4 | | | | Compare M:M+1 from D | | + | + | + | + |
| | CMPS | 11 | 5 | 4 | 11 | 7 | 3 | 11 | 7+ | 3+ | 11 | 8 | 4 | | | | Compare M:M+1 from S | | + | + | + | + |

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 8C | | | 9C | | | AC | | | BC | | | | | | | | | |
| | CMPU | 11 83 | 5 | 4 | 11 93 | 7 | 3 | 11 A3 | 7+ | 3+ | 11 B3 | 8 | 4 | | Compare M:M+1 from U | | + | + | + | + |
| | CMPX | 8C | 4 | 3 | 9C | 6 | 2 | AC | 6+ | 2+ | BC | 7 | 3 | | Compare M:M+1 from X | | + | + | + | + |
| | CMPY | 10 8C | 5 | 4 | 10 9C | 7 | 3 | 10 AC | 7+ | 3+ | 10 BC | 8 | 4 | | Compare M:M+1 from Y | | + | + | + | + |
| COM | COMA | | | | | | | | | | | | | 43 | 2 | 1 | A = complement(A) | | + | + | 0 | 1 |
| | COMB | | | | | | | | | | | | | 53 | 2 | 1 | B = complement(B) | | + | + | 0 | 1 |
| | COM | | | | 03 | 6 | 2 | 63 | 6+ | 2+ | 73 | 7 | 3 | | | | M = complement(M) | | + | + | 0 | 1 |
| CWAI | | 3C | => 20 | 2 | | | | | | | | | | | | | CC = CC ^ IMM; Wait for Interrupt | | | | | 7 |
| DAA | | | | | | | | | | | | | | 19 | 2 | 1 | Decimal Adjust A | | + | + | 0 | + |
| DEC | DECA | | | | | | | | | | | | | 4A | 2 | 1 | A = A 1 | | + | + | + | |
| | DECB | | | | | | | | | | | | | 5A | 2 | 1 | B = B 1 | | + | + | + | |
| | DEC | | | | 0A | 6 | 2 | 6A | 6+ | 2+ | 7A | 7 | 3 | | | | M = M 1 | | + | + | + | |
| EOR | EORA | 88 | 2 | 2 | 98 | 4 | 2 | A8 | 4+ | 2+ | B8 | 5 | 3 | | | | A = A XOR M | | + | + | 0 | |
| | EORB | C8 | 2 | 2 | D8 | 4 | 2 | E8 | 4+ | 2+ | F8 | 5 | 3 | | | | B = M XOR B | | + | + | 0 | |
| EXG | R1,R2 | 1E | 8 | 2 | | | | | | | | | | | | | exchange R1,R2 | | | | | |
| INC | INCA | | | | | | | | | | | | | 4C | 2 | 1 | A = A + 1 | | + | + | + | |
| | INCB | | | | | | | | | | | | | 5C | 2 | 1 | B = B + 1 | | + | + | + | |
| | INC | | | | 0C | 6 | 2 | 6C | 6+ | 2+ | 7C | 7 | 3 | | | | M = M + 1 | | + | + | + | |
| JMP | | | | | 0E | 3 | 2 | 6E | 3+ | 2+ | 7E | 4 | 3 | | | | pc = EA | | | | | |
| JSR | | | | | 9D | 7 | 2 | AD | 7+ | 2+ | BD | 8 | 3 | | | | jump to subroutine | | | | | |

6809 Instruction Set

| Instruction | Mnemonic | Addressing Mode | | | | | | | | | | | | | | | Description | CC bit | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Immediate | | | Direct | | | Indexed | | | Extended | | | Inherent | | | | 5 | 3 | 2 | 1 | 0 |
| | | Op | ~ | # | Op | ~ | # | Op | ~ | # | Op | ~ | # | Op | ~ | # | | H | N | Z | V | C |
| LD | LDA | 86 | 2 | 2 | 96 | 4 | 2 | A6 | 4+ | 2+ | B6 | 5 | 3 | | | | A = M | | + | + | 0 | |
| | LDB | C6 | 2 | 2 | D6 | 4 | 2 | E6 | 4+ | 2+ | F6 | 5 | 3 | | | | B = M | | + | + | 0 | |

| | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LDD | CC | 3 | 3 | DC | 5 | 2 | EC | 5+ | 2+ | FC | 6 | 3 | | | | D = M:M+1 | | + | + | 0 | |
| | LDS | 10 CE | 4 | 4 | 10 DE | 6 | 3 | 10 EE | 6+ | 3+ | 10 FE | 7 | 4 | | | | S = M:M+1 | | + | + | 0 | |
| | LDU | CE | 3 | 3 | DE | 5 | 2 | EE | 5+ | 2+ | FE | 6 | 3 | | | | U = M:M+1 | | + | + | 0 | |
| | LDX | 8E | 3 | 3 | 9E | 5 | 2 | AE | 5+ | 2+ | BE | 6 | 3 | | | | X = M:M+1 | | + | + | 0 | |
| | LDY | 10 8E | 4 | 4 | 10 9E | 6 | 3 | 10 AE | 6+ | 3+ | 10 BE | 7 | 4 | | | | Y = M:M+1 | | + | + | 0 | |
| LEA | LEAS | | | | | | | 32 | 4+ | 2+ | | | | | | | S = EA | | | | | |
| | LEAU | | | | | | | 33 | 4+ | 2+ | | | | | | | U = EA | | | | | |
| | LEAX | | | | | | | 30 | 4+ | 2+ | | | | | | | X = EA | | | + | | |
| | LEAY | | | | | | | 31 | 4+ | 2+ | | | | | | | Y = EA | | | + | | |
| LSL | LSLA | | | | | | | | | | | | | 48 | 2 | 1 | Logical shift left | | + | + | + | + |
| | LSLB | | | | | | | | | | | | | 58 | 2 | 1 | | | + | + | + | + |
| | LSL | | | | 08 | 6 | 2 | 68 | 6+ | 2+ | 78 | 7 | 3 | | | | | | + | + | + | + |
| LSR | LSRA | | | | | | | | | | | | | 44 | 2 | 1 | Logical shift right | | 0 | + | | + |
| | LSRB | | | | | | | | | | | | | 54 | 2 | 1 | | | 0 | + | | + |
| | LSR | | | | 04 | 6 | 2 | 64 | 6+ | 2+ | 74 | 7 | 3 | | | | | | 0 | + | | + |
| MUL | | | | | | | | | | | | | | 3D | 11 | 1 | D = A*B (Unsigned) | | | + | | 9 |
| NEG | NEGA | | | | | | | | | | | | | 40 | 2 | 1 | A = !A + 1 | 8 | + | + | + | + |
| | NEGB | | | | | | | | | | | | | 50 | 2 | 1 | B = !B + 1 | 8 | + | + | + | + |
| | NEG | | | | 00 | 6 | 2 | 60 | 6+ | 2+ | 70 | 7 | 3 | | | | M = !M + 1 | 8 | + | + | + | + |
| NOP | | | | | | | | | | | | | | 12 | 2 | 1 | No Operation | | | | | |
| OR | ORA | 8A | 2 | 2 | 9A | 4 | 2 | AA | 4+ | 2+ | BA | 5 | 3 | | | | A = A \|\| M | | + | + | 0 | |
| | ORB | CA | 2 | 2 | DA | 4 | 2 | EA | 4+ | 2+ | FA | 5 | 3 | | | | B = B \|\| M | | + | + | 0 | |
| | ORCC | 1A | 3 | 2 | | | | | | | | | | | | | C = CC \|\| IMM | ? | ? | ? | ? | ? |
| PSH | PSHS | 34 | 5+ | 2 | | | | | | | | | | | | | Push Registers on S Stack | | | | | |
| | PSHU | 36 | 5+ | 2 | | | | | | | | | | | | | Push Registers on U Stack | | | | | |
| PUL | PULS | 35 | 5+ | 2 | | | | | | | | | | | | | Pull Registers from S Stack | | | | | |
| | PULU | 37 | 5+ | 2 | | | | | | | | | | | | | Pull Registers from U Stack | | | | | |
| ROL | ROLA | | | | | | | | | | | | | 49 | 2 | 1 | Rotate left thru carry | | + | + | + | + |

| | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ROLB | | | | | | | | | | | | 59 | 2 | 1 | | | + | + | + | + |
| | ROL | | | | 09 | 6 | 2 | 69 | 6+ | 2+ | 79 | 7 | 3 | | | | | + | + | + | + |
| ROR | RORA | | | | | | | | | | | | 46 | 2 | 1 | Rotate Right thru carry | | 0 | + | | + |
| | RORB | | | | | | | | | | | | 56 | 2 | 1 | | | 0 | + | | + |
| | ROR | | | | 06 | 6 | 2 | 66 | 6+ | 2+ | 76 | 7 | 3 | | | | | 0 | + | | + |
| RTI | | | | | | | | | | | | | 3B | 6/15 | 1 | Return from Interrupt | ? | ? | ? | ? | ? |
| RTS | | | | | | | | | | | | | 39 | 5 | 1 | Return from subroutine | | | | | |
| SBC | SBCA | 82 | 2 | 2 | 92 | 4 | 2 | A2 | 4+ | 2+ | B2 | 5 | 3 | | | A = A - M - C | 8 | + | + | + | + |
| | SBCB | C2 | 2 | 2 | D2 | 4 | 2 | E2 | 4+ | 2+ | F2 | 5 | 3 | | | B = B - M - C | 8 | + | + | + | + |
| SEX | | | | | | | | | | | | | 1D | 2 | 1 | Sign extend B into A | | + | + | 0 | |
| ST | STA | | | | 97 | 4 | 2 | A7 | 4+ | 2+ | B7 | 5 | 3 | | | M = A | | + | + | 0 | |
| | STB | | | | D7 | 4 | 2 | E7 | 4+ | 2+ | F7 | 5 | 3 | | | M = B | | + | + | 0 | |
| | STD | | | | DD | 5 | 2 | ED | 5+ | 2+ | FD | 6 | 3 | | | M:M+1 = D | | + | + | 0 | |
| | STS | | | | 10 DF | 6 | 3 | 10 EF | 6+ | 3+ | 10 FF | 7 | 4 | | | M:M+1 = S | | + | + | 0 | |
| | STU | | | | DF | 5 | 2 | EF | 5+ | 2+ | FF | 6 | 3 | | | M:M+1 = U | | + | + | 0 | |
| | STX | | | | 9F | 5 | 2 | AF | 5+ | 2+ | BF | 6 | 3 | | | M:M+1 = X | | + | + | 0 | |
| | STY | | | | 10 9F | 6 | 3 | 10 AF | 6+ | 3+ | 10 BF | 7 | 4 | | | M:M+1 = Y | | + | + | 0 | |
| SUB | SUBA | 80 | 2 | 2 | 90 | 4 | 2 | A0 | 4+ | 2+ | B0 | 5 | 3 | | | A = A - M | 8 | + | + | + | + |
| | SUBB | C0 | 2 | 2 | D0 | 4 | 2 | E0 | 4+ | 2+ | F0 | 5 | 3 | | | B = B - M | 8 | + | + | + | + |
| | SUBD | 83 | 4 | 3 | 93 | 6 | 2 | A3 | 6+ | 2+ | B3 | 7 | 3 | | | D = D - M:M+1 | | + | + | + | + |
| SWI | SWI | | | | | | | | | | | | 3F | 19 | 1 | Software interrupt 1 | | | | | |
| | SWI2 | | | | | | | | | | | | 10 3F | 20 | 2 | Software interrupt 2 | | | | | |
| | SWI3 | | | | | | | | | | | | 11 3F | 20 | 2 | Software interrupt 3 | | | | | |
| SYNC | | | | | | | | | | | | | 13 | >= 4 | 1 | Synchronize to Interrupt | | | | | |
| TFR | R1,R2 | 1F | 6 | 2 | | | | | | | | | | | | R2 = R1 | | | | | |
| TST | TSTA | | | | | | | | | | | | 4D | 2 | 1 | Test A | | + | + | 0 | |

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TSTB | | | | | | | | | 5D | 2 | 1 | Test B | | + | + | 0 | |
| | TST | | 0D | 6 | 2 | 6D | 6+ | 2+ | 7D | 7 | 3 | | Test M | | + | + | 0 | |

:

| Legend: | | |
|---|---|---|
| ! Complement of M | + Test and set if true, cleared otherwise | OP Operation Code(Hexadecimal) |
| = Transfer from | - Not Affected | ~ Number of MPU Cycles |
| H Half carry (from bit 3) | CC Condition Code Register | # Number of Program Bytes |
| N Negative (sign bit) | : Concatenation | + Arithmetic Plus |
| Z Zero (Reset) | \|\| Logical or | Arithmetic Minus |
| V Overflow, 2's complement | && Logical and | * Multiply |
| C Carry from ALU | EOR Logical Exclusive or | EA Effective Address:w |

**Notes:**

1. This column gives a base cycle and byte count. To obtain total count, add the values obtained from the INDEXED ADDRESSING MODE table, in Appendix F.

2. Rl and R2 may be any pair of 8 bit or any pair of 16 bit registers.
The 8 bit registers are: A, B, CC, DP
The 16 bit registers are: X, Y, U, S, D, PC

3. EA is the effective address.

4. The PSH and PUL instructions require 5 cycles plus 1 cycle for each byte pushed or pulled.

5. 5(6) means: 5 cycles if branch not taken, 6 cycles if taken (Branch instructions.

6. SWI sets I and F bits. SW12 and SW13 do not affect I and F.

7. Conditions Codes set as a direct result of the instruction.

8. Value of half carry flag is undefined.

9. Special Case Carry set if b7 is SET.

| Instruction | Forms | Mode Relstive OP | Decription | 5 H | 3 N | 2 Z | 1 V | O C |
|---|---|---|---|---|---|---|---|---|
| BCC | BCC | 24 3 / | 2 Branch C=O | | | | | |
| | LBCC | 10 516) | 4 Long Branch | | | | | |
| | | 24 | C=O | | | | | |
| BCS | BCS | 25 3 | 2 Branch C= 1 | | | | | |
| | LBCS | 10 56) | 4 Long Branch | | | | | |
| | | 25 | C=1 | | | | | |
| BEQ | BEQ | 27 3 | 2 Branch Z=O | | | | | |
| | LBEQ | 10 5(6) | 4 Long Branch | | | | | |
| | | 27 | Z=O | | | | | |
| BGE | BGE | 2C 3 | 2 Branch2Zero | | | | | |
| | LBGE | 10 5(6) | 4 Long Branch2Zero | | | | | |
| | | 2C | | | | | | |
| BGT | BGT | 2E 3 | 2 Branch > Zero | | | | | |
| | LBGT | 10 5(6) | 4 Long Branch>Zero | | | | | |
| | | 2E | | | | | | |
| BHI | BHI | 22 3 | 2 Branch rligher | | | | | |
| | LBHI | 10 5(6) | 4 Long Branch Higher | | | | | |
| | | 22 | | | | | | |
| BHS | BHS | 24 3 | 2 Branch Higher or Same | | | | | |
| | LBHS | 10 516) | 4 Long Branch Higher | | | | | |
| | | 24 | or Same | | | | | |
| BLE | BLE | 2F 3 | 2 BranchsZero | | | | | |
| | LBLE | 10 5(6) | 4 Long BranchsZero | | | | | |
| | | 2F | | | | | | |
| BLO | BLO | 25 3 | 2 Branch lower | | | | | |

```
             LBLO    10 56)    4 Long Branch Lower
                     25
```

```
    Addressin        T
       Mode
      Rela         5 3 2 1 ,0
Instruction Forms OP   #    Description H N Z V C

BLS     BLS   23 3     2 Branch Lower
                         or Same
        LBLS  10 5(6)  4 Long Branch Lower
              23         or Same

BLT     BLT   2D 3     2 Branch
```