MC6809 COURSE OUTLINE

- INTRODUCTION/OVERVIEW
- PROGRAMMING MODEL
- SIGNAL AND CONTROL LINES
- INSTRUCTION SET
- ADDRESSING MODES
- COMPARISONS WITH OTHER 68XX'S
- MACRO ASSEMBLER
- PROGRAMMING EXAMPLES
- PROBLEM SET
M6800 Microprocessor/Microcomputer Family Evolution

A CONSISTENT, COMPATIBLE FAMILY

MC6809

MPU

16 BIT ADR BUS

8 BIT DATA BUS

CONTROL BUS

MCM6810

RAM

128 x 8

MCM6830

ROM

1024 x 8

MC6821

"A"

"B"

- PIA

- 8 BITS

PARALLEL DATA

MC6850

ACIA

Tx

Rx

SERIAL DATA

8 BITS
CONDITION CODE REGISTER

- Carry Borrow
- Overflow (1’s Comp.)
- Zero Result
- Negative (b = 1, N = 1)
- Interrupt Mask (IRQ)
- Half Carry (b3 = b0)
- Fast Interrupt Mask (FIRO)
- Entire Machine Status Stacked

BITS SET AS A RESULT OF OPERATION!

<table>
<thead>
<tr>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDA A = 1000 1000, M = 1000 1000</td>
<td>A + M = 1 0001 0000</td>
</tr>
<tr>
<td>DEC A = 0000 0001</td>
<td>A-1 = 0000 0000</td>
</tr>
<tr>
<td>LDA #A50</td>
<td>A = 1000 0000</td>
</tr>
<tr>
<td>COMA A = 1000 0000</td>
<td>A = 0111 1111</td>
</tr>
<tr>
<td>OR DECA</td>
<td>A = 1000 0010 = -126&lt;sub&gt;10&lt;/sub&gt;</td>
</tr>
<tr>
<td>ADDA A = 1000 0010, M = 1000 0010</td>
<td>A + M = 1 0000 0100</td>
</tr>
</tbody>
</table>

MC6809 BUS & CONTROL SIGNALS

+5 V GND

BUS AVAILABLE
HALT
BUS STATUS
IRQ
NON-MASKABLE INT.
RESET
DMA REQ
MEMORY READY
XTAL
EXTAL
IRQ

MC6809

+8

DATA BUS

16

ADDRESS BUS

CONTROL BUS

EOUT
QOUT
READ/WRITE
MC6809
MREADY
(MEMORY READY)

\[ \begin{align*}
\text{Eout} & \quad \text{\{} \quad \text{\{} \\
\text{Qout} & \quad \text{\{} \quad \text{\{} \\
\text{MREADY} & \quad \text{\{} \quad \text{\{} \\
\end{align*} \]

\text{MREADY LOW} \leq 10\mu s

MC6809
DMAREQ

- STRETCHES INTERNAL CLOCK
- EOUT AND QOUT FREE RUN
- BA = 1; BS = 1
- ADDRESS BUS, DATA BUS, R/W GO THREE STATE
- EVERY 16 CYCLES 6809 WILL EXECUTE ONE CYCLE
- BA = 0 FOR THIS CYCLE. AUTOMATIC MPU REFRESH
- BS = 0
**RESTART SEQUENCE**

- Signal received on restart pin
- X1X1XXXX → CCR
  00 → DPR (Mask interrupts)
- Load program counter from (FFFF) → PC
  (0000) → PC
- Go to location determined by PC and begin initialization

Diagram:
- Vcc
- RES
- 4.75 V
- 5 clock cycles
- Reset uses Schmidt trigger
- +5 V
- 10K
- .1 μF
### Hardware Interrupt

<table>
<thead>
<tr>
<th></th>
<th>1 (Bit 4 of CCR)</th>
<th>F (Bit 5 of CCR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NM</td>
<td>SET</td>
<td>SET</td>
</tr>
<tr>
<td>IRQ</td>
<td>SET</td>
<td>SET</td>
</tr>
<tr>
<td>SWI</td>
<td>SET</td>
<td>NO CHANGE</td>
</tr>
<tr>
<td>SWI</td>
<td>NO CHANGE</td>
<td>NO CHANGE</td>
</tr>
<tr>
<td>SWI</td>
<td>NO CHANGE</td>
<td>NO CHANGE</td>
</tr>
</tbody>
</table>

#### Stack MPU Register Contents

- **V** (Non Mask)
- **N** (Mask Set)

- **Set Mask 1 (F)**

#### Load Interrupt Vector into Program Counter

- **Condition Code**
- **Accumulator B**
- **Accumulator A**

#### Set Mask 1 & F

**IRO** only sets the flag SWI 2 & SWI 3 do not set flags (I, F)

---

**Program Counter:**

- **SP-1**
- **SP-2**
- **SP-3**
- **SP-4**
- **SP-5**
- **SP-6**
- **SP-7**
- **SP-8**
- **SP-9**
- **SP-10**

**Stack Condition Code Reg.:**

- **PC (L)**
- **PC (H)**
- **SP (L)**
- **SP (H)**

---

**Legend:**

- **RESTART (L):** FFFF
- **RESTART (H):** FFFD
- **NON MASKABLE (L):** FFDC
- **NON MASKABLE (H):** FFDB
- **SOFTWARE (L):** FFFA
- **SOFTWARE (H):** FFFA
- **HARDWARE (L):** FF9
- **HARDWARE (H):** FF9
- **FAST IRQ (L):** FF6
- **FAST IRQ (H):** FF6
- **SOFTWARE 2 (L):** FF4
- **SOFTWARE 2 (H):** FF4
- **SOFTWARE 3 (L):** FF3
- **SOFTWARE 3 (H):** FF2
RAM USED FOR STACK STORAGE
RAM ADDRESS $0000 - $007F

HARDWARE STACK POINTER AT START OF SEQUENCE

GOOD DATA
PC LOW
PC HIGH
US LOW
US HIGH
Y LOW
Y HIGH
X LOW
X HIGH
DPR
Acc B
Acc A
CCR

ENDING POINTER (AFTER INTERRUPT)

007F

0073

0000
interrupt acknowledge (IACK) function

- IACK goes high to indicate a vector is being fetched
- This signal could be used to activate address decoding logic
- Therefore, generating new vectors
- BS = 1; BA = 0
MC6809E BUS & CONTROL SIGNALS
BASIC EXTERNAL CLOCK GENERATOR FOR MC6809E

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>JA</th>
<th>KA</th>
<th>JB</th>
<th>KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

2XFC OSC.

(A)

(B)

J

K

Q

Q

EIN

QIN

START OF CYCLE ¼ CYCLE ½ CYCLE ¾ CYCLE END OF CYCLE

EIN

QIN

TSC

MC6809E

- ADDRESS; R/W GO THREE STATE
- DATA BUS GOES THREE STATE
- BA = 0; BS = 0
- CLOCK MUST BE HELD LOW EXTERNALLY.
MC6809E
NEW CONTROL LINES

LAST INSTRUCTION CYCLE (LIC)
- LIC GOES HIGH AT THE BEGINNING OF THE LAST INSTRUCTION CYCLE
- NEXT CYCLE WILL BE AN OP CODE FETCH
- USED TO INDICATE WHEN TO READ BUS FOR DIAGNOSTIC PURPOSES

MC6809E
NEW CONTROL LINES
BUSY

- GOES HIGH DURING READ-MODIFY-WRITE INSTRUCTIONS
- GOES HIGH DURING DOUBLE BYTE OPERATIONS
- GOES HIGH DURING INDIRECT OPERATIONS
- BUSY = 0 FOR CWAI
- BUSY = 0 FOR HALT
- BUSY IS A COMMON BUS LOCKOUT SIGNAL
SUMMARY
CONTROL LINE STATES

<table>
<thead>
<tr>
<th>BA</th>
<th>DMA REQ*</th>
<th>CWA1</th>
<th>TSC</th>
<th>SYNC</th>
<th>MRDY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>LIC</td>
<td>1</td>
<td>NA</td>
<td>1</td>
<td>1/0</td>
<td>1</td>
</tr>
<tr>
<td>BS</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BUSY</td>
<td>0</td>
<td>NA</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

ENABLE OUT

QUAD OUT

VMA

DATA BUS

ADD BUS R/W

TS | TS | ACTIVE | TS | TS | ACTIVE

*EVERY 18 CYCLES, DMA REQ AUTOMATICALLY refreshes 6809, BA = 0

STATES SIGNALS

<table>
<thead>
<tr>
<th>EA</th>
<th>BS</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>NORMAL OPERATION</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>INTERRUPT ACKNOWLEDGE</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>SYNC ACKNOWLEDGE</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>DMA GRANT OR HALT ACKNOWLEDGE</td>
</tr>
</tbody>
</table>

1K-BYTE ROM

ADDRESSES

FD60 | INTERUPT ROUTINES
FD10 | INTERUPT (NMI)
FC80 | INTERUPT (SWI)
FOAO | INTERUPT (SWI)
FC10 | EXECUTIVE CODE
FC10 | INTERRUPT ROUTINES (IRQ)
FD10 | INTERRUPT ROUTINES (IRQ)
FD10 | INTERRUPT ROUTINES (IRQ)
## M6809

### 59 INSTRUCTIONS

#### 9 ADDRESSING MODES

---

**ARITHMETIC INSTRUCTIONS**

<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>FUNCTION</th>
<th>MNEMONIC</th>
<th>OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>ADD ACCUMULATOR TO MEMORY</td>
<td>ADDA</td>
<td>A + (M) → A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDS</td>
<td>B + (M) → B</td>
</tr>
<tr>
<td></td>
<td>ADD DOUBLE ACCUMULATOR TO MEMORY</td>
<td>ADDDD</td>
<td>Dₜ + (M) → Dₜ</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDD</td>
<td>Dₜ + (M + 1) → Dₜ</td>
</tr>
<tr>
<td>ADDC</td>
<td>ADD ACCUMULATOR WITH CARRY TO MEMORY</td>
<td>ADDCA</td>
<td>A + (M) + C → A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDCB</td>
<td>B + (M) + C → B</td>
</tr>
<tr>
<td>ABX</td>
<td>ADD B ACCUMULATOR TO THE X REGISTER</td>
<td>ABX</td>
<td>B + Xₜ → Xₜ</td>
</tr>
<tr>
<td></td>
<td>DAX</td>
<td>DAA</td>
<td>Converts binary add. of BCD characters into BCD format</td>
</tr>
<tr>
<td>SUB</td>
<td>SUBTRACT MEMORY FROM ACCUMULATOR</td>
<td>SUBA</td>
<td>A - (M) → A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SUBB</td>
<td>B - (M) → B</td>
</tr>
<tr>
<td></td>
<td>SUBTRACT MEMORY FROM DOUBLE ACCUMULATORS</td>
<td>SBBDA</td>
<td>Dₜ - (M + 1) → Dₜ</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SBBD</td>
<td>Dₜ - M → Dₜ</td>
</tr>
<tr>
<td>SBC</td>
<td>SUBTRACT MEMORY FROM ACCUMULATOR WITH CARRY</td>
<td>SBCA</td>
<td>A - (M) - C → A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SBCB</td>
<td>B - (M) - C → B</td>
</tr>
<tr>
<td>SEX</td>
<td>EXTEND THE SIGN OF B INTO A</td>
<td>SEX</td>
<td></td>
</tr>
</tbody>
</table>

---
## DATA HANDLING INSTRUCTIONS
(ALTERNATE DATA)

<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>FUNCTION</th>
<th>MNEMONIC</th>
<th>OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CLEAR MEMORY</td>
<td>CLR</td>
<td>CO → M</td>
</tr>
<tr>
<td></td>
<td>CLEAR Acc A</td>
<td>CLRA</td>
<td>CO → A</td>
</tr>
<tr>
<td></td>
<td>CLEAR Acc B</td>
<td>CLRb</td>
<td>CO → B</td>
</tr>
<tr>
<td>DEC</td>
<td>DECREMENT MEMORY</td>
<td>DEC</td>
<td>(M) - 1 → M</td>
</tr>
<tr>
<td></td>
<td>DECREMENT Acc A</td>
<td>DECA</td>
<td>A - 1 → A</td>
</tr>
<tr>
<td></td>
<td>DECREMENT Acc B</td>
<td>DECB</td>
<td>B - 1 → B</td>
</tr>
<tr>
<td>INC</td>
<td>INCREMENT MEMORY</td>
<td>INC</td>
<td>(M) + 1 → M</td>
</tr>
<tr>
<td></td>
<td>INCREMENT Acc A</td>
<td>INCa</td>
<td>A + 1 → A</td>
</tr>
<tr>
<td></td>
<td>INCREMENT Acc B</td>
<td>INCb</td>
<td>B + 1 → B</td>
</tr>
<tr>
<td>COM</td>
<td>COMPLEMENT MEMORY</td>
<td>COM</td>
<td>(M) → M</td>
</tr>
<tr>
<td></td>
<td>COMPLEMENT Acc A</td>
<td>COMA</td>
<td>A → A</td>
</tr>
<tr>
<td></td>
<td>COMPLEMENT Acc B</td>
<td>COMB</td>
<td>B → B</td>
</tr>
<tr>
<td>NEG</td>
<td>2'S COMPLEMENT MEMORY</td>
<td>NEG</td>
<td>CO → (M) → M</td>
</tr>
<tr>
<td></td>
<td>2'S COMPLEMENT Acc A</td>
<td>Nega</td>
<td>CO → A → A</td>
</tr>
<tr>
<td></td>
<td>2'S COMPLEMENT Acc B</td>
<td>NEGb</td>
<td>CO → B → B</td>
</tr>
</tbody>
</table>

## DATA HANDLING INSTRUCTIONS
(DATA MOVEMENT)

<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>FUNCTION</th>
<th>MNEMONIC</th>
<th>OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD</td>
<td>LOAD Acc.</td>
<td>LDA</td>
<td>(M) → A</td>
</tr>
<tr>
<td></td>
<td>LOAD 16 BIT REG</td>
<td>LDA</td>
<td>(M) → B</td>
</tr>
<tr>
<td></td>
<td>LOAD 16 BIT REG</td>
<td>LDD</td>
<td>(M) → Rₘ</td>
</tr>
<tr>
<td></td>
<td>LOAD 16 BIT REG</td>
<td>LUX</td>
<td>(M + 1) → Rₘ</td>
</tr>
<tr>
<td></td>
<td>LOAD 16 BIT REG</td>
<td>LOY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LOAD 16 BIT REG</td>
<td>LDS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LOAD 16 BIT REG</td>
<td>LDU</td>
<td></td>
</tr>
<tr>
<td>ST</td>
<td>STORE Acc</td>
<td>STA</td>
<td>A → M</td>
</tr>
<tr>
<td></td>
<td>STORE 16 BIT REG</td>
<td>STB</td>
<td>B → M</td>
</tr>
<tr>
<td></td>
<td>STORE 16 BIT REG</td>
<td>STD</td>
<td>Rₘ → M</td>
</tr>
<tr>
<td></td>
<td>STORE 16 BIT REG</td>
<td>STX</td>
<td>Rₘ → M + 1</td>
</tr>
<tr>
<td></td>
<td>STORE 16 BIT REG</td>
<td>STY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>STORE 16 BIT REG</td>
<td>STS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>STORE 16 BIT REG</td>
<td>STJ</td>
<td></td>
</tr>
</tbody>
</table>
### DATA HANDLING INSTRUCTIONS
#### (DATA MOVEMENT)

<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>FUNCTION</th>
<th>MNEMONIC</th>
<th>OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSH</td>
<td>PUSH REGISTER(S) TO HARDWARE STACK</td>
<td>PSHS, PSHU</td>
<td>SP - 1 → SP, USP - 1 → USP, R → Msp, USP</td>
</tr>
<tr>
<td></td>
<td>PUSH REGISTER(S) TO USER STACK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PUL</td>
<td>PULL REGISTER(S) FROM HARDWARE STOCK</td>
<td>PULS, PULU</td>
<td>(Msp) → R, SP + 1 → SP, (Msp) → R, USP + 1 → USP</td>
</tr>
<tr>
<td></td>
<td>PULL REGISTER(S) FROM USER STACK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TFR</td>
<td>TRANSFER REGISTER TO REGISTER</td>
<td>TFR</td>
<td>R₁ ↔ R₂</td>
</tr>
<tr>
<td>EXG</td>
<td>EXCHANGE REGISTERS</td>
<td>EXG</td>
<td>R₁ ↔ R₂</td>
</tr>
</tbody>
</table>

---

### DATA HANDLING INSTRUCTIONS
#### (SHIFT AND ROTATE)

<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>FUNCTION</th>
<th>MNEMONIC</th>
<th>OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROL</td>
<td>ROTATE LEFT</td>
<td>ROL, ROLA, ROLR</td>
<td><img src="TR0040-1" alt="ROL Diagram" /></td>
</tr>
<tr>
<td>ROR</td>
<td>ROTATE RIGHT</td>
<td>ROR, RORA, RORB</td>
<td><img src="TR0040-1" alt="ROR Diagram" /></td>
</tr>
<tr>
<td>LSL (ASL)</td>
<td>LOGICAL SHIFT LEFT</td>
<td>LSL, LSLA, LSLB</td>
<td><img src="TR0040-1" alt="LSL Diagram" /></td>
</tr>
<tr>
<td>ASR</td>
<td>ARITHMETIC SHIFT RIGHT</td>
<td>ASR, ASRA, ASRB</td>
<td><img src="TR0040-1" alt="ASR Diagram" /></td>
</tr>
<tr>
<td>LSR</td>
<td>LOGICAL SHIFT RIGHT</td>
<td>LSR, LSRA, LSRB</td>
<td><img src="TR0040-1" alt="LSR Diagram" /></td>
</tr>
</tbody>
</table>
### Logic Instructions

<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>FUNCTION</th>
<th>MNEMONIC</th>
<th>OPERATION</th>
</tr>
</thead>
</table>
| AND         | AND ACCUMULATOR WITH MEMORY | ANDA, ANDB | A • (M) → A  
B • (M) → B |
| EOR         | EXCLUSIVE OR ACCUMULATOR WITH MEMORY | EORA, EORB | A ⊕ (M) → A  
B ⊕ (M) → B |
| OR          | INCLUSIVE OR ACCUMULATOR WITH MEMORY | ORA, ORA | A + (M) → A  
B + (M) → B |

### Data Test Instructions

<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>FUNCTION</th>
<th>MNEMONIC</th>
<th>TEST</th>
</tr>
</thead>
</table>
| BIT         | BIT TEST (AND) ACCUMULATOR WITH MEMORY | BITA, BITB | A • (M)  
B • (M) |
| CMP         | COMPARE (SUBTRACT) MEMORY WITH ACCUMULATOR | CMPA, CMPB | A − (M)  
B − (M) |
|             | COMPARE (SUBTRACT) MEMORY WITH 15 BIT REGISTER | CMPD, CMPX, CMPY, CMPU, CMPS | R_e − (M + 1)  
R_f − (M − 1) |
| TST         | TEST MEMORY FOR ZERO OR MINUS TEST ACCUMULATOR FOR ZERO OR MINUS | TST, TSTA, TSTB | (M) − 00  
A − 00  
B − 00 |
## BRANCH INSTRUCTIONS

<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>FUNCTION</th>
<th>MNEMONIC</th>
<th>BRANCH TEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRA</td>
<td>BRANCH ALWAYS</td>
<td>BRA</td>
<td>NONE</td>
</tr>
<tr>
<td>BRN</td>
<td>BRANCH NEVER</td>
<td>BRN</td>
<td>NONE</td>
</tr>
<tr>
<td>BCC (BHS)</td>
<td>BRANCH IF CARRY CLEAR (HIGHER OR SAME)</td>
<td>BCC</td>
<td>C = 0</td>
</tr>
<tr>
<td>BCS (BLO)</td>
<td>BRANCH IF CARRY SET (IF LOWER)</td>
<td>BCS</td>
<td>C = 1</td>
</tr>
<tr>
<td>GEQ</td>
<td>BRANCH IF = ZERO</td>
<td>GEQ</td>
<td>Z = 1</td>
</tr>
<tr>
<td>GNE</td>
<td>BRANCH IF NOT EQUAL ZERO</td>
<td>GNE</td>
<td>Z = 0</td>
</tr>
<tr>
<td>BGE</td>
<td>BRANCH IF ≥ ZERO*</td>
<td>BGE</td>
<td>(N + V) = 0</td>
</tr>
<tr>
<td>BLT</td>
<td>BRANCH IF &lt; ZERO*</td>
<td>BLT</td>
<td>(N + V) = 1</td>
</tr>
<tr>
<td>BGT</td>
<td>BRANCH IF &gt; ZERO*</td>
<td>BGT</td>
<td>Z + (N + V) = 0</td>
</tr>
<tr>
<td>BGE</td>
<td>BRANCH IF ≥ ZERO*</td>
<td>BGE</td>
<td>Z + (N + V) = 1</td>
</tr>
<tr>
<td>BLE</td>
<td>BRANCH IF &lt; ZERO*</td>
<td>BLE</td>
<td>(C + Z) = 0</td>
</tr>
<tr>
<td>BHI</td>
<td>BRANCH IF HIGHER</td>
<td>BHI</td>
<td>(C + Z) = 1</td>
</tr>
<tr>
<td>BLS</td>
<td>BRANCH IF LOWER OR SAME</td>
<td>BLS</td>
<td>N = 1</td>
</tr>
<tr>
<td>BMI</td>
<td>BRANCH IF MINUS</td>
<td>BMI</td>
<td>N = 0</td>
</tr>
<tr>
<td>BPL</td>
<td>BRANCH IF PLUS</td>
<td>BPL</td>
<td></td>
</tr>
</tbody>
</table>

* SIGNED DATA ONLY!

## JUMP AND BRANCH INSTRUCTIONS

<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>FUNCTION</th>
<th>MNEMONIC</th>
<th>BRANCH TEST/OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BVC</td>
<td>BRANCH IF OVERFLOW CLEAR</td>
<td>BVC</td>
<td>V = 0</td>
</tr>
<tr>
<td>BVS</td>
<td>BRANCH IF OVERFLOW SET</td>
<td>BVS</td>
<td>V = 1</td>
</tr>
<tr>
<td>BSR</td>
<td>BRANCH TO SUBROUTINE</td>
<td>BSR</td>
<td>NONE</td>
</tr>
<tr>
<td>JMP</td>
<td>JUMP TO MEMORY ADDRESS</td>
<td>JMP</td>
<td>(M) → PC; (M + 1) → PC</td>
</tr>
<tr>
<td>JSR</td>
<td>JUMP TO SUBROUTINE MEMORY ADDRESS</td>
<td>JSR</td>
<td>SP → SP; PC → M</td>
</tr>
<tr>
<td>RTS</td>
<td>RETURN FROM SUBROUTINE</td>
<td>RTS</td>
<td>(M) → PC; (M + 1) → PC, SP + 1 → SP</td>
</tr>
</tbody>
</table>
## CONDITION CODE REGISTER INSTRUCTION

<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>FUNCTION</th>
<th>MNEMONIC</th>
<th>OPERATION</th>
</tr>
</thead>
</table>
| OR          | SET ANY CONDITION CODE FLAG | ORCC | \[
\begin{align*}
CC + (M) & \rightarrow CC \\
E & \quad F \quad H \quad I \quad N \quad Z \quad V \quad C \\
0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 \\
+ & \quad & & & & & \quad M = 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \\
E & \quad F \quad H \quad I \quad N \quad Z \quad V \quad C \\
0 & \quad 0 & \quad 0 & \quad 0 & \quad 1 & \quad 1 & \quad 1 
\end{align*}
\]  |
| AND         | CLEAR ANY CONDITION CODE FLAG | ANDCC | \[
\begin{align*}
CC + (M) & \rightarrow CC \\
E & \quad F \quad H \quad I \quad N \quad Z \quad V \quad C \\
0 & \quad 0 & \quad 0 & \quad 0 & \quad 1 & \quad 1 & \quad 1 \\
+ & \quad & & & & & \quad M = 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \\
E & \quad F \quad H \quad I \quad N \quad Z \quad V \quad C \\
0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 
\end{align*}
\]  |

## INTERRUPT HANDLING INSTRUCTION

<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>FUNCTION</th>
<th>MNEMONIC</th>
<th>OPERATION</th>
</tr>
</thead>
</table>
| RTI         | RETURN FROM INTERRUPT | RTI | \[
(M_{IR}) & \rightarrow CC \\
SP + 1 & \rightarrow SP \\
IF E = 1 & \\
(M_{IR}) & \rightarrow REGS \\
SP + B & \rightarrow SP \\
IF E = 0 & \\
(M_{IR}) & \rightarrow P_{IR} \\
SP + 1 & \rightarrow SP \\
M_{IR} & \rightarrow P_{IR} \\
SP + 1 & \rightarrow SP 
\]  |
| CWAI        | CLEAR AND WAIT FOR INTERRUPT | CWAI | \[
CC \cdot M \\
E = 1 \\
REGS & (M_{IR}) \\
SP - C & \rightarrow SP 
\]  |
| SYNC        | SYNCHRONIZE TO INTERRUPT | SYNC | \begin{align*}
\text{STOPS MPU UNTIL INTERRUPT OCCURS THEN CONTINUES}
\end{align*}  |
### INTERRUPT HANDLING INSTRUCTION

<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>FUNCTION</th>
<th>MNEMONIC</th>
<th>OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW1</td>
<td>SOFTWARE INTERRUPT</td>
<td>SWI</td>
<td>E = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>REGS → M trương</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SP – C → SP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>F = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(FFFA) → P advant.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(FFFD) → P cân</td>
</tr>
<tr>
<td>SW2</td>
<td></td>
<td>SWI</td>
<td>E = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>REGS → M trương</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SP – C → SP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(FFF4) = P cân</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(FFF5) = P cân</td>
</tr>
<tr>
<td>SW3</td>
<td></td>
<td>SWI</td>
<td>E = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>REGS → M trương</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SP – C → SP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(FFF2) = P cân</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(FFF3) = P cân</td>
</tr>
</tbody>
</table>

### POINTER REGISTER INSTRUCTIONS

<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>FUNCTION</th>
<th>MNEMONIC</th>
<th>OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEA</td>
<td>LOAD POINTER REGISTER WITH EFFECTIVE ADDRESS</td>
<td>LEAX</td>
<td>EA → X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LEAY</td>
<td>EA → Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LEAS</td>
<td>EA → S</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LEAU</td>
<td>EA → U</td>
</tr>
</tbody>
</table>

Note only uses indexed addressing mode
# Miscellaneous Instruction

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Function</th>
<th>Mnemonic</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>MUL</td>
<td>Multiply Accumulators</td>
<td>MUL</td>
<td>A * B → D</td>
</tr>
<tr>
<td>NOP</td>
<td>No Operation</td>
<td>NOP</td>
<td>PC + 1 → PC</td>
</tr>
</tbody>
</table>

**Input/Output Instructions Still None!**

---

TR9052
### 6809 EXECUTABLE INSTRUCTIONS — ALPHABETIC LIST

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>ADD ACROSS</td>
</tr>
<tr>
<td>ADDA</td>
<td>ADD WITH CARRY</td>
</tr>
<tr>
<td>AND</td>
<td>AND LOGICAL</td>
</tr>
<tr>
<td>ASL</td>
<td>ARITHMETIC SHIFT LEFT (LSL)</td>
</tr>
<tr>
<td>ASR</td>
<td>ARITHMETIC SHIFT RIGHT</td>
</tr>
<tr>
<td>BCC</td>
<td>BRANCH IF CARRY CLEAR</td>
</tr>
<tr>
<td>BCS</td>
<td>BRANCH IF CARRY SET</td>
</tr>
<tr>
<td>BNE</td>
<td>BRANCH IF NOT EQUAL</td>
</tr>
<tr>
<td>BPL</td>
<td>BRANCH IF PLUS</td>
</tr>
<tr>
<td>BPS</td>
<td>BRANCH IF PROGRAM</td>
</tr>
<tr>
<td>BRK</td>
<td>BRANCH TO SUBROUTINE</td>
</tr>
<tr>
<td>BSR</td>
<td>BRANCH TO SUBROUTINE</td>
</tr>
<tr>
<td>CLC</td>
<td>CLEAR</td>
</tr>
<tr>
<td>CMP</td>
<td>COMPLEMENT</td>
</tr>
<tr>
<td>CMI</td>
<td>CLEAR AND WAIT FOR INT</td>
</tr>
<tr>
<td>DEC</td>
<td>DECREMENT</td>
</tr>
<tr>
<td>EXC</td>
<td>EXCLUSIVE OR</td>
</tr>
<tr>
<td>EXG</td>
<td>EXCHANGE REGISTERS</td>
</tr>
<tr>
<td>INX</td>
<td>INCREMENT</td>
</tr>
<tr>
<td>JMP</td>
<td>JUMP</td>
</tr>
<tr>
<td>JSR</td>
<td>JUMP TO SUBROUTINE</td>
</tr>
<tr>
<td>LD</td>
<td>LOAD ACUMULATORS</td>
</tr>
<tr>
<td>LD</td>
<td>LOAD POINTER REGISTERS</td>
</tr>
<tr>
<td>LEA</td>
<td>LOAD EFFECTIVE ADDRESS</td>
</tr>
<tr>
<td>LSR</td>
<td>LOGICAL SHIFT RIGHT</td>
</tr>
<tr>
<td>MUL</td>
<td>MULTIPLY ACUMULATORS</td>
</tr>
<tr>
<td>NEG</td>
<td>NEGATE</td>
</tr>
<tr>
<td>NOP</td>
<td>NO OPERATION</td>
</tr>
<tr>
<td>OR</td>
<td>OR INCLUSIVE OR ACUMULATORS/CONDITION CODE REGISTER</td>
</tr>
<tr>
<td>PHS</td>
<td>PUSH DATA TO HARDWARE OR USER STACK</td>
</tr>
<tr>
<td>PUL</td>
<td>PULL DATA FROM HARDWARE OR USER STACK</td>
</tr>
<tr>
<td>ROL</td>
<td>ROTATE LEFT</td>
</tr>
<tr>
<td>ROR</td>
<td>ROTATE RIGHT</td>
</tr>
<tr>
<td>RTS</td>
<td>RETURN FROM INTERRUPT</td>
</tr>
<tr>
<td>SBC</td>
<td>SUBTRACT WITH CARRY</td>
</tr>
<tr>
<td>SEX</td>
<td>SIGN EXTENDED</td>
</tr>
<tr>
<td>SNI</td>
<td>STORE ACUMULATORS</td>
</tr>
<tr>
<td>ST</td>
<td>STORE POINTER REGISTER</td>
</tr>
<tr>
<td>SUB</td>
<td>SUBTRACT</td>
</tr>
<tr>
<td>SWI</td>
<td>SOFTWARE INTERRUPT</td>
</tr>
<tr>
<td>SWD</td>
<td>SOFTWARE INTERRUPT</td>
</tr>
<tr>
<td>STS</td>
<td>SOFTWARE INTERRUPT</td>
</tr>
<tr>
<td>TFR</td>
<td>TRANSFER REGISTERS</td>
</tr>
<tr>
<td>TST</td>
<td>TEST FOR ZERO</td>
</tr>
</tbody>
</table>

### 6809 ADDRESSING MODES

- Immediate
- Inherent
- Direct
- Extended
- Indexed
- Relative
- Extended Indirect
- Indexed Indirect
- Long Relative
SPECIAL CASES OF INHERENT ADDRESSING MODE

EXCHANGE, TRANSFER REGISTERS

ExG   X, Y
1E    12 (X ← Y)

OPCODE REGISTER

<table>
<thead>
<tr>
<th>HEX CHAR.</th>
<th>HEX CHAR.</th>
</tr>
</thead>
<tbody>
<tr>
<td>REG. 1</td>
<td>REG. 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HEX CHAR.</th>
<th>REG.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>Y</td>
</tr>
<tr>
<td>3</td>
<td>U</td>
</tr>
<tr>
<td>4</td>
<td>S</td>
</tr>
<tr>
<td>5</td>
<td>PC</td>
</tr>
<tr>
<td>6</td>
<td>A</td>
</tr>
<tr>
<td>7</td>
<td>B</td>
</tr>
<tr>
<td>8</td>
<td>CC</td>
</tr>
<tr>
<td>9</td>
<td>DP</td>
</tr>
</tbody>
</table>
• PULL AND PULL

PULS A, B, X

35 16 (00010110)

OPCODE REGISTERS

PULS #STACK

POST BYTE

BIT POSITION

B7 B6 B5 B4 B3 B2 B1 B0
PC U/S Y X D B A C

REGISTERS
PUSHED OR PULLED

• CWAI

CWAI #$FF

3C FF

OPCODE POST BYTE

CWAI #MASKI

CONDITION
CODE REG.

FF • EFHI "NZVC

TR8115

TR8115-1
IMMEDIATE ADDRESSING

GENERAL FLOW

MPU

MEMORY

DATA

OP CODE

IMMEDIATE
TWO BYTE
INSTRUCTION

EXAMPLE

MPU

MEMORY

Acc A

25

EXAMPLE

LDA #25

0002

0001

IMMEDIATE ADDRESSING

GENERAL FLOW

MPU

MEMORY

DATA

OP CODE

IMMEDIATE
THREE BYTE
INSTRUCTION

EXAMPLE

LDX #2526

X REG

[25 26]

26

25

8E

5003

5002

5001
IMMEDIATE ADDRESSING

GENERAL FLOW
MPU
MEMORY
DATA LOW
DATA HIGH
OP CODE
PRE BYTE

IMMEDIATE
FOUR BYTE
INSTRUCTION

EXAMPLE
MPU
MEMORY
Y-REG
28 28
26
25
8E
10
5003
5002
5001
5000

DIRECT ADDRESSING

GENERAL FLOW
MPU
MEMORY
DPR
ADDR HIGH

DIRECT
TWO BYTE
INSTRUCTION

EXAMPLE
MPU
MEMORY
DPR.
00
Acc A
3A

EXAMPLE
LDA <SF2
ADDR LOW
0 - SFF
FA

MEMORY
5002
5001
96
3A
50F2
**Relative Addressing**

**General Flow**
- MPU
- MEMORY
  - NEXT OP CODE
  - PC
  - NEXT OP CODE
  - PC
  - RELATIVE
  - TWO BYTE
  - \(-128 \leq \text{OFFSET} \leq +127\)
  - $800 - 0 - 7FF$

**Example**
- MPU
- MEMORY
  - CCR
  - Z
  - PC
  - NEXT OP CODE
  - 6002
  - 6001
  - FO
  - 27

**Long Relative Addressing**

**General Flow**
- MPU
- MEMORY
  - NEXT OP CODE
  - PC
  - NEXT OP CODE
  - PC
  - PRE BYTE

- "**LONG RELATIVE**
  - FOur BYTE
  - INSTRUCTION"
- \(-32,768 \leq \text{OFFSET} \leq +32,767\)
- \$8000 - 0 - 7FFF$

**Example**
- MPU
- MEMORY
  - CCR
  - Z
  - PC
  - NEXT OP CODE
  - 6004
  - 6003
  - 00
  - 27
  - FO
  - 19

**Notes:**
- LBRA and LBSR have their own op code; therefore, these instructions require no pre-byte and are 3 bytes long.

30
INDEXED ADDRESSING
(8-BIT OFFSET)

GENERAL FLOW
MPU

MEMORY
OFFSET
POST BYTE
OP CODE

INDEXED
EIGHT BIT OFFSET
THREE BYTE
INSTRUCTION

-128 < OFFSET < +127
$00 - 0 - $7F

EXAMPLE
MPU

MEMORY
X-REG
1A00
Acc A
D9

POST BYTE
1 0 0 0 1 0 0 0
X-REG FETCH
ONE BYTE OFFSET

INDEXED ADDRESSING
(16-BIT OFFSET)

GENERAL FLOW
MPU

MEMORY
OFFSET LOW
OFFSET HIGH
POST BYTE
OP CODE

INDEXED
SIXTEEN BIT OFFSET
FOUR BYTE
INSTRUCTION

-32,768 < OFFSET < +32,767
$8000 - 0 - $7FFF

EXAMPLE
MPU

MEMORY
X-REG
1A00
Acc A
D9

POST BYTE
1 0 0 0 0 1 0 0 1
X-REG 16 BIT
OFFSET

EXAMPLE
MPU

MEMORY
X-REG
65
5003

D9
5001
1A65
INDEXED ADDRESSING
(8-BIT OFFSET TO PC)

INDEXED EIGHT BIT OFFSET
THREE BYTE INSTRUCTION
\[-128 \leq \text{OFFSET} \leq +127\]
\$00 - 0 - \$7F
*OFFSET TO PC

INDEXED ADDRESSING
(16-BIT OFFSET TO PC)

INDEXED SIXTEEN BIT OFFSET
FOUR BYTE INSTRUCTION
\[-32768 \leq \text{OFFSET} \leq +32767\]
\$8000 - 0 - \$7FFFF
*OFFSET TO PC
INDEXED ADDRESSING
(ACCUMULATOR OFFSET A, B OR D)

GENERAL FLOW

MPU

MEMORY

INDEXED
ACCUMULATOR
OFFSET

TWO BYTE
INSTRUCTION

-128 < OFFSET < +127

80H — 0 — 8FH

16 BIT OFFSET FOR D ACCUMULATOR

MPU

MEMORY

EXAMPLE

LDA B,X

POST BYTE

10001010  01

X-REG  D ALL

OFFSET

MEMORY

EXAMPLE

1A05

D9

5002

5001

INDEXED ADDRESSING
(AUTO INC/DEC)

GENERAL FLOW

MPU

MEMORY

INDEXED

AUTO INC/DEC

TWO BYTE
INSTRUCTION

NO OFFSET

INC/DEC = 1 OR -2

MPU

MEMORY

EXAMPLE

LDA ,X++

POST BYTE

10000000  01

X-REG  AUTO

INC +2

MEMORY

EXAMPLE

B3

S002

S001

1A00
INDEXED ADDRESSING
(AUTO INC/DEC)

GENERAL FLOW

MPU

MEMORY

PRE BYTE

OP CODE

POST BYTE

INDEXED

AUTO INC/DEC

THREE BYTE

INSTRUCTION

NO OFFSET

AUTO INC/DEC = 1 OR = 2

EXAMPLE

MPU

MEMORY

X-REG
IA06 $10FE

Y-REG
5221

INDEXED

AUTO INC/DEC

THREE BYTE

INSTRUCTION

NO OFFSET

AUTO INC/DEC = 1 OR = 2

EXAMPLE

STY, -X

POST BYTE

10 00 00 01 11

X-REG AUTO

-2

VALID INDEX MODE COMBINATIONS

<table>
<thead>
<tr>
<th>OPERAND REGISTER</th>
<th>X</th>
<th>Y</th>
<th>S</th>
<th>U</th>
<th>PCR</th>
<th>- - - X</th>
<th>- - - Y</th>
<th>- - - S</th>
<th>- - - U</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFFSET</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Constant + or - up to 16 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. OPERATION REGISTER MAY BE A, B, D, X, Y, S, OR U.
2. SHARED AREAS ARE INVALID COMBINATIONS.
3. WHEN PCR IS THE OPERAND REGISTER, THE NUMBER IN THE OFFSET
   POSITION OF THE ASSEMBLY LANGUAGE STATEMENT IS THE MEMORY
   LOCATION ADDRESS OR LABEL. THE ASSEMBLER CALCULATES THE OFFSET.
### 6809 Indexed Addressing Modes

<table>
<thead>
<tr>
<th>Type</th>
<th>NON INDIRECT</th>
<th>INDIRECT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ASSEMBLER FORM</td>
<td>POST-BYTE OPCODE</td>
</tr>
<tr>
<td>CONSTANT OFFSET FROM R</td>
<td>a, R</td>
<td>1RR0000 0 0</td>
</tr>
<tr>
<td>NO OFFSET</td>
<td>a, R</td>
<td>1RR0100 0 0</td>
</tr>
<tr>
<td>5 BIT OFFSET</td>
<td>a, R</td>
<td>1RR00100 1 1</td>
</tr>
<tr>
<td>8 BIT OFFSET</td>
<td>a, R</td>
<td>1RR01010 2 2</td>
</tr>
<tr>
<td>16 BIT OFFSET</td>
<td>a, R</td>
<td>1RR01011 4 0</td>
</tr>
<tr>
<td>ACCUMULATOR OFFSET FROM R</td>
<td>A, R</td>
<td>1RR0000 1 0</td>
</tr>
<tr>
<td>A-REGISTER OFFSET</td>
<td>D, R</td>
<td>1RR01011 4 0</td>
</tr>
<tr>
<td>B-REGISTER OFFSET</td>
<td>B, R</td>
<td>1RR01010 2 0</td>
</tr>
<tr>
<td>D-REGISTER OFFSET</td>
<td>D, R</td>
<td>1RR01011 4 0</td>
</tr>
<tr>
<td>AUTO INCREMENT/DECREMENT R</td>
<td>R+</td>
<td>1RR0000 2 0</td>
</tr>
<tr>
<td>INCREMENT BY 1</td>
<td>R++</td>
<td>1RR0001 3 0</td>
</tr>
<tr>
<td>INCREMENT BY 2</td>
<td>R-</td>
<td>1RR01010 2 0</td>
</tr>
<tr>
<td>DECREMENT BY 1</td>
<td>R-</td>
<td>1RR01011 4 0</td>
</tr>
<tr>
<td>DECREMENT BY 2</td>
<td>R-</td>
<td>1RR01011 4 0</td>
</tr>
<tr>
<td>CONSTANT OFFSET FROM PC</td>
<td>a, PCR</td>
<td>1XX0000 3 1</td>
</tr>
<tr>
<td>8 BIT OFFSET</td>
<td>a, PCR</td>
<td>1XX00010 2 2</td>
</tr>
<tr>
<td>16 BIT OFFSET</td>
<td>a, PCR</td>
<td>1XX00011 4 0</td>
</tr>
</tbody>
</table>

**Notes:****
- RR = 00 = X
- 01 = Y
- 10 = U
- 11 = S

### Extended Indirect Addressing

**GENERAL FLOW**

**MEMORY**

**MPU**

**EXAMPLE**

**MEMORY**

**Example:**
- LDA [2A34] 
- OP CODE FOR LDA INDEXED
INDEXED INDIRECT ADDRESSING
(NO OFFSET)

GENERAL FLOW

MPU

MEMORY

MPU

EXAMPLE

MEMORY

INDEXED INDIRECT
NO OFFSET

TWO BYTE
INSTRUCTION

EXAMPLE

LDA [X]

POST BYTE
100 10100

X-REG
NO OFFSET
INDIRECT

B7

2040

INDEXED INDIRECT
ADDRESSING
(8-BIT OFFSET)

GENERAL FLOW

MPU

MEMORY

MPU

EXAMPLE

MEMORY

INDEXED INDIRECT
EIGHT BIT OFFSET

THREE BYTE
INSTRUCTION

-128 ≤ OFFSET ≤ 127

500 0 ≤ 57F

EXAMPLE

LDA [$55,X]

POST BYTE
100 11000

X-REG
EIGHT BIT
OFFSET
INDIRECT

7C

4080

65

90

A6

5002

5001

1A00

1A01

1A03

5000

5002

5001

1A66

1A65
INDEXED INDIRECT ADDRESSING
(16-BIT OFFSET)

GENERAL FLOW

MPU

MEMORY

OFFSET LOW
OFFSET HIGH
POST BYTE
OP CODE

INDEXED INDIRECT
SIXTEEN BIT OFFSET
FOUR BYTE
INSTRUCTION

-32,768 \leq \text{OFFSET} \leq +32,767
S8000 = 0 \rightarrow S7FFF

EXAMPLE

MPU

MEMORY

X-REG
0A00
Acc A
47

EXAMPLE

LDA (S2000,X)
POST BYTE
1B91 10 01
X-REG 16-BIT
OFFSET
INDIRECT

-3722

2A01
2A00

INDEXED INDIRECT ADDRESSING
(8-BIT OFFSET TO PC)

GENERAL FLOW

MPU

MEMORY

OFFSET
POST BYTE
OP CODE

INDEXED INDIRECT
RIGHT BIT OFFSET
THREE BYTE
INSTRUCTION

-128 \leq \text{OFFSET} \leq +127
S80 = 0 \rightarrow S7F
OFFSET TO PC

EXAMPLE

MPU

MEMORY

PC
5000
Acc A
27

EXAMPLE

LDA (S4FF4, PC)
POST BYTE
1B91 11 10 00
DONT CARE
PC+8 BIT
OFFSET
INDIRECT

4FF5
4FF4

5003
5002
5001

35
20
2035
INDEXED INDIRECT ADDRESSING (16-BIT OFFSET TO PC)

GENERAL FLOW

MPU

MEMORY

OFFSET LOW
OFFSET HIGH
OP CODE
POST BYTE

EXAMPLE

MPU

MEMORY

PC
[5003]
Acc A
1F

EXAMPLE

LDA [54005, PC]
POST BYTE
1 0 0 1 1 1 0 1
DON'T CARE
PC-16
BIT OFFSET
INDIRECT

INDEXED INDIRECT
SIXTEEN BIT OFFSET
FOUR BYTE INSTRUCTION
-32768 < OFFSET < +32767
$8000 - 0 — 877FF

*OFFSET TO PC

INDEXED INDIRECT ADDRESSING
(ACCUMULATOR OFFSET A, B OR D)

GENERAL FLOW

MPU

MEMORY

POST BYTE
OP CODE

EXAMPLE

MPU

MEMORY

X-REG
IA00
Acc B
06
Acc A
B3

EXAMPLE

LDA (B, X)
POST BYTE
1 0 0 1 0 1 0 1
X-REG
Acc B
OFFSET
INDIRECT

INDEXED INDIRECT
ACCUMULATOR OFFSET
TWO BYTE INSTRUCTION
-128 < OFFSET < +127
$80 — 0 — 57F
*16 BIT OFFSET
FOR D ACCUMULATOR
INDEXED INDIRECT ADDRESSING
(AUTO INC/DEC = 2)

GENERAL FLOW

MPU

MEMORY

INDEXED INDIRECT
AUTO INC/DEC
TWO BYTE
INSTRUCTION
NO OFFSET
AUTO INC/DEC = 2

EXAMPLE

MPU

MEMORY

EXAMPLE
LDA (X++)
POST BYTE
1.0.0.1.0.0.1
X-REG AUTO INC +2
INDIRECT

5002
91
A5

2A07
C0

5001
1A31

07
1A00

1011
### SUMMARY OF PRE-BYTE RELATED INSTRUCTIONS

<table>
<thead>
<tr>
<th>PRE-BYTE</th>
<th>PRE-BYTE</th>
<th>PRE-BYTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWI</td>
<td>SWI2</td>
<td>SWI3</td>
</tr>
<tr>
<td>SUBD</td>
<td>CMPD</td>
<td>CMPS</td>
</tr>
<tr>
<td>CMPX</td>
<td>CMPY</td>
<td>—</td>
</tr>
<tr>
<td>LDX</td>
<td>LDY</td>
<td>—</td>
</tr>
<tr>
<td>STX</td>
<td>STY</td>
<td>—</td>
</tr>
<tr>
<td>LDU</td>
<td>LDS</td>
<td>—</td>
</tr>
<tr>
<td>STU</td>
<td>STS</td>
<td>—</td>
</tr>
<tr>
<td>BRANCH</td>
<td><em>LONGBRANCH</em></td>
<td>—</td>
</tr>
</tbody>
</table>

*EXCEPT FOR BRA, LBRA, BSR, AND LBSR*
## MC6800/01/02/09 COMPARISONS

<table>
<thead>
<tr>
<th></th>
<th>6800</th>
<th>VIA</th>
<th>6802</th>
<th>6803</th>
<th>6809</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLOW</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
</tr>
<tr>
<td>MEMORY</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
</tr>
<tr>
<td>INPUT</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
</tr>
<tr>
<td>31-12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>27-0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>STACK</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
</tr>
<tr>
<td>POINT</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
</tr>
<tr>
<td>OPERATIONS</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
</tr>
<tr>
<td>INTERRUPT</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
</tr>
<tr>
<td>CONDITION</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
</tr>
<tr>
<td>REGISTERS</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
</tr>
<tr>
<td>INTERRUPT</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
</tr>
<tr>
<td>CONDITION</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
</tr>
<tr>
<td>REGISTERS</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
</tr>
<tr>
<td>13 BIT</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
</tr>
<tr>
<td>COMPARE</td>
<td>FOR 128</td>
<td>FOR 128</td>
<td>FOR 128</td>
<td>FOR 128</td>
<td>FOR 128</td>
</tr>
<tr>
<td>CONDITION</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
</tr>
<tr>
<td>REGISTERS</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
</tr>
<tr>
<td>A8A</td>
<td>ASH</td>
<td>ASH</td>
<td>ASH</td>
<td>ASH</td>
<td>ASH</td>
</tr>
<tr>
<td>A8H</td>
<td>ASH</td>
<td>ASH</td>
<td>ASH</td>
<td>ASH</td>
<td>ASL</td>
</tr>
<tr>
<td>A8L</td>
<td>ASH</td>
<td>ASH</td>
<td>ASH</td>
<td>ASH</td>
<td>ASL</td>
</tr>
<tr>
<td>A8S</td>
<td>ASH</td>
<td>ASH</td>
<td>ASH</td>
<td>ASH</td>
<td>ASL</td>
</tr>
</tbody>
</table>

---

**PROGRAMS WRITTEN FOR THE 6800/01 HAVE AN INCREASED CHANCE OF RUNNING ON THE 6809 IF THE FOLLOWING CONDITIONS ARE MET:**

- **PROGRAM MUST BE REASSEMBLED (ALL OP CODES ARE NOT THE SAME)**
- **ONLY STANDARD USE OF THE STACK (I.E. NOTHING OTHER THAN SIMPLE INTERRUPTS AND SUBROUTINE CALLS)**
- **ADDRESS FFF0 → FFF7 NOT USED**
- **SOFTWARE TIMING LOOPS NOT CRITICAL (DIFFERENT CYCLE TIMES)**
- **DATA WRITTEN TO/FROM CCR CAN NOT ASSUME HIGHER ORDER BITS EQUAL TO ONES**
SET DIRECT PAGE DIRECTIVE
+ THIS PROGRAM SHOWS THE EFFECT OF
+THE DIRECT PAGE DIRECTIVE IN ASSEMBLY
+OF A PROGRAM.

000001     NAM  DIPGGE
000002     SETDP  $1A
000003     LDA  $1A00
000004     LDA  $1B00
000005     LDA  $1A00
000006     001A  A
000007     0000  9E  00
000008     0002  9E  1B00  A
000009     0005  9E  00  A
000010     LDA  <$1A00

+++WARNING 002--0000
00010A     0007  9E  00  A
00011A     0009  9E  1A00  A
00012A     000C  9E  1B00  A
00013A     000F  9E  FF  A
00014A     001B  A
00015A     001A  9E  00  A
00016A     LDA  $1B00
00017A     LDA  >$1A00
00018A     LDA  >$1B00
00019A     LDX  <$1AFF
0001AA     001B  A
0001BA     SETDP  $1B
00020A     LDA  $1B00
00021A     END

TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00001--00010
16 X 16 MULTIPLY ROUTINE

\[
\begin{align*}
3344 & \quad 2344 \\
1122 & \quad 1122 \\
6688 & \quad 08 \\
6688 & \quad 08 \quad 44 \times 22 = 908 \\
3344 & \quad 66 \quad 33 \times 22 = 6C6 \\
6688 & \quad 44 \quad 44 \times 11 = 484 \\
33 & \quad 33 \times 11 = 363 \\
036E5308 & \quad 036E5308
\end{align*}
\]

V1H V1L
V1 = 33 44
V2 = 11 22
V2H V2L
PAGE 001

00011
00020
00029
00038
00047
00056
00065
00074
00083
00092
00101
00110
00119
00128
00137
00146
00155
00164
00173
00182
00191
00200
00209
00218
00227
00236
00245
00254
00263
00272
00281
00290
00300
00309
00318
00327
00336
00345
00354
00363
00372
00381
00390
00400

PAGE 002

00409
00418
00427
00436
00445
00454
00463
00472
00481
00490
00500
00509
00518
00527
00536
00545
00554
00563
00572
00581
00590
00600
00609
00618
00627
00636
00645
00654
00663
00672
00681
00690
00700
00709
00718
00727
00736
00745
00754
00763
00772
00781
00790
00800
00809
00818
00827
00836
00845
00854
00863
00872
00881
00890
00900
00909
00918
00927
00936
00945
00954
00963
00972
00981
00990

1616 MULTIPLE-RESULTANT
MULTIPLE THE 16-BIT POSITIVE VALUES
TO GENERATE A 32-BIT PRODUCT.

PRODUCT

V1=16-BIT FIRST VARIABLE
V2=16-BIT SECOND VARIABLE
V1XV2 = 32-BIT PRODUCT

V1=16-BIT FIRST VARIABLE
V2=16-BIT SECOND VARIABLE
V1XV2 = 32-BIT PRODUCT
PERIPHERAL POLLING

- STRAIGHT-LINE METHOD RESULTS IN FASTEST POLLING ROUTINE; ALSO NORMALLY USES THE MOST MEMORY.
- MINIMUM INSTRUCTION ROUTINE MEANS MORE TIME REQUIRED FOR POLLING.
- DEPENDING UPON USER CONSTRAINTS SUCH AS NUMBER OF PERIPHERALS, INTERRUPT RESPONSE TIME, AVAILABLE MEMORY, ETC., EITHER TYPE OF ROUTINE CAN BE USED, OR PERHAPS A COMBINATION OF THE TWO WILL BE OPTIMUM.