**Addressing Modes of 6800**

* Inherent (Implied)
* Direct /Extended
* Immediate
* Indexed
* Relative

**Inherent**

Instructions, which do not have any operand OR their Operand is predefined in the OPCODE (Usually Accumulator). These instructions take 1 byte in the Memory. For Example:

NOP (No Operation)

CLR A (Clear Accumulator A)

SUB B (B = B-1)

CLC (Clear Carry Flag)

PSH A (Push A into Stack)

**Immediate**

In this mode, Operand itself is specified in the program memory location, next to the OPCODE

These instructions occupy two bytes of the program memory.

Byte 1: OPCODE

Byte 2: Operand

**Direct**

It is special case of Extended Addressing Mode.

In the Direct Addressing Mode, Only LSB of Address is specified. The MSB of the address is assumed as ZERO. These take only Two bytes in the memory.

Hence only ZERO PAGE of the memory address space can be accessed.

**Extended**

In Extended Addressing Mode, Address of the Operand is specified in the program memory locations, next to the OPCODE. These Instructions take three bytes in the memory.

Byte 1: OPCODE

Byte 2: MSB of Operand’s Address

Byte 3: LSB of Operand’s Address

**Indexed**

These Instructions specify operands address with the help of Index Registers.

The Index Register provided Base Address, which is added with an offset to determine the operand’s address.

These Instructions take two bytes in program memory:

BYTE 1: OPCODE

BYTE 2: Offset

**Relative**

It is used only with branching instructions. The microprocessor jumps to a location relative to its present location.

These instructions occupy two bytes of program memory:

Byte 1: OPCODE

Byte 2: Offset

The offset is added into the contents of Program Counter. If offset is positive, the jump is Forward; otherwise, the jump is Backwards.

**6800 Address Modes**

There are 6 different address modes on the Motorola 6800 microprocessor.

**Implied:**

If the instruction is implied (eg. inca) then the contents of Accumulator A are incremented by 1 and the fetch-instruction action is complete. This instruction operates with the internal registers only.

**Immediate:**

 If the instruction is immediate (eg. ldaa #$02) then the next byte is fetched  and treated as data, in this example 2 is placed  into the Accumulator A.

**Direct:**

If the instruction is direct (eg. ldaa $42) then the next byte is fetched and 00 is placed as the High Byte of  the address ((00)42) of the data to be acted upon. So it can only be used with addresses from $0000 to $00FF.

**Extended:**

If the instruction is extended (eg. ldaa $C300) then the next two bytes are fetched at the address ($C300)  and the data to be acted upon are the contents of this address. So the contents of address ($C300) will be placed into Accumulator A.

**Indexed:**

If the instruction is Indexed then the contents of the Index register are added to the Displacement byte ($00 to $FF) and this forms a pointer to the contents of the data to be acted upon.

**Relative:**

If the instruction is relative then the next byte is fetched  and treated as an offset to the PC during the following execution. This offset is a [Signed Number](http://www.electronics.dit.ie/staff/tscarff/signed_numbers/signed_numbers.htm) and allows the programme to branch forward up to +127 steps and backwards to a maximum of -128 steps.

The final phase is the execution cycle. This depends on what the instruction is. For example if it is staa 42h than it will take the contents of the Accumulator out to the Data register and onto the Data bus, will place the address (00)42h on the Address bus and bring the R/W line low to write the data into the memory at the specified address.