ECE 3120
Computer Systems
Addressing Modes

Manjeera Jeedigunta

http://blogs.cae.tntech.edu/msjeedigun21

Email: msjeedigun21@tntech.edu

Tel: 931-372-6181, Prescott Hall 120

Addressing Modes

- How CPU accesses the memory locations?
  - A HCS12 instruction consists of one or two bytes of opcode and zero to five bytes of operand addressing information.
    - Opcode bytes specify the operation to be performed by the CPU.
    - The first byte of a two-byte opcode is always $18.
- Addressing modes specify the operand to be operated on.
- The addressing mode may specify a value, a register, or a memory location to be used as an operand.

Effective address**: memory address affected by the instruction.**

Table 1.2P M68HC12 addressing mode summary

Addressing mode	Source format	Abbre.	Description
Inherent	**INST** (no externally supplied operands)	INH	Operands (if any) are in CPU registers
Immediate	**INST** #opr8i or **INST** #opr16i	IMM	Operand is included in instruction stream. 8- or 16-bit size implied by context
Direct	**INST** opr8a	DIR	Operand is the lower 8 bits of an address in the range $0000-$00FF
Extended	**INST** opr16a	EXT	Operand is a 16-bit address
Relative	**INST** rel8 or **INST** rel16	REL	An 8-bit or 16-bit relative offset from the current PC is supplied in the instruction
Indexed (5-bit offset)	**INST** oprx5,xysp	IDX	5-bit signed constant offset from x,y,sp, or pc
Indexed (pre-decrement)	**INST** oprx3,-xys	IDX	Auto pre-decrement x, y, or sp by 1 ~ 8
Indexed (pre-increment)	**INST** oprx3,+xys	IDX	Auto pre-increment x, y, or sp by 1 ~ 8
Indexed (post-decrement)	**INST** oprx3,xys-	IDX	Auto post-decrement x, y, or sp by 1 ~ 8
Indexed (post-increment)	**INST** oprx3,xys+	IDX	Auto post-increment x, y, or sp by 1 ~ 8
Indexed (accumulator offset)	**INST** abd,xysp	IDX	Indexed with 8-bit (A or B) or 16-bit (D) accumulator offset from x, y, sp, or pc
Indexed (9-bit offset)	**INST** oprx9,xysp	IDX1	9-bit signed constant offset from x, y, sp, or pc (lower 8-bits of offset in one extension byte)
Indexed (16-bit offset)	**INST** oprx16,xysp	IDX2	16-bit constant offset from x, y, sp, or pc (16-bit offset in two extension bytes)
Indexed-Indirect (16-bit offset)	**INST** [oprx16,xysp]	[IDX2]	Pointer to operand is found at 16-bit constant offset from (x, y, sp, or pc)
Indexed-Indirect (D accumulator offset)	**INST** [**D**,xysp]	[D,IDX]	Pointer to operand is found at x, y, sp, or pc plus the vlaue in D

Inherent Mode

- Instructions that use this mode do not use extra bytes to specify operands because the instructions either do not need operands or all operands are CPU registers.
  - Operands are implied by the opcode.
  - Examples
    - NOP
    - INX
    - DECA

Immediate Mode

- Operands for instructions that use immediate mode are included in the instruction.
- CPU does not access memory for operands.
- Example
  - LDAA #$55
  - LDX #$1000

Direct Mode

- This mode can only specify memory locations in the range of 0 - 255.
- This mode uses only one byte to specify the operand address.
- Example
  - LDAA $20
  - LDAB $40

Extended Mode

- In this mode, the full 16-bit address is provided in the instruction.
- For example,
  - LDAA $4000
  - LDX$FE60

Relative Mode (1 of 2)

- Used only by branch instructions
- Short and long conditional branch instructions use exclusively relative mode.
- BRCLR and BRSET instructions can also use relative mode to specify branch target.
  - A short branch instructions consists of an 8-bit opcode and a signed 8-bit offset.
  - The short relative mode can specify a range of -128 ~ +127.
  - A long branch instruction consists of an 8-bit opcode and a signed 16-bit offset.
  - The range of the long relative mode is from -32768 ~ +32767.
    - A programmer uses a symbol to specify the branch target and the assembler will figure out the actual branch offset (distance) from the instruction that follows branch instruction.

Relative Mode (2 of 2)

- For example,

  minus

  …

  …

  bmi minus

  …

Indexed Mode

- This mode uses the sum of an index register (X, Y, PC, or SP) and an offset to specify the address of an operand.
  - The offset can be a 5-bit, 9-bit, and 16-bit signed value or the value in accumulator A, B, or D.
  - Automatic pre- or post-increment or pre- or post-decrement by -8 to +8 are options.
  - PC can be used as the index register for all but auto-increment or auto-decrement mode.
  - Indirect indexing with 16-bit offset or accumulator D as the offset is supported.
    - A summary of indexed addressing modes is given in Table 1.3.

Table 1.3 Summary of indexed operations

Postbyte code (xb)	source code syntax	**Comments** rr: 00 = X, 01 = Y, 10 = SP, 11 = PC
rr0nnnnn	r	**5-bit constant** offset n = -16 to +15
	n,r	r can be X, Y, SP, or PC
	-n,r	
111rr0zs	n,r	**Constant offset** (9- or 16-bit signed)
	-n,r	z: 0 = 9-bit with sign in LSB of postbyte (s) -256< n < 255
		1 = 16-bit 0 < n < 65535
		if z = s = 1, 16-bit offset indexed-indirect (see below)
		r can be X, Y, SP, or PC
111rr011	[n,r]	**16-bit offset indexed-indirect** 0 < n < 65536
		rr can be X, Y, SP, or PC
rr1pnnnn	n,-r	**Auto pre-decrement/increment or auto post-decrement/increment**;
	n,+r	p = pre-(0) or post-(1), n = -8 to -1 or +1 to +8
	n,r-	r can be X, Y, or SP (PC not a valid choice)
	n,r+	+8 = 0111
		...
		+1 = 0000
		-1 = 1111
		....
		-8 = 1000
111rr1aa	A,r	**Accumulator offset** (unsigned 8-bit 0r 16-bit)
	B,r	aa: 00 = A
	D,r	01 = B
		10 = D (16-bit)
		11 = see accumulator D offset indexed-indirect
		r can be X, Y, SP, or PC
111rr111	[D,r]	**Accumulator D offset indexed-indirect**
		r can be X, Y, SP, or PC

Indexed Addressing (1 of 2)

- 5-bit Constant Offset Indexed Addressing
  - The base index register can be X, Y, SP, or PC.
  - The range of the offset is from -16 to +15.
  - Examples
    - ldaa 0,X
    - stab -8,0
- 9-bit Constant Offset Indexed Addressing
  - The base index register can be X, Y, SP, or PC.
  - The range of the offset is from -256 to +255.
  - Examples
    - ldaa $FF,X
    - ldab -20,Y

Indexed Addressing (2 of 2)

- 16-bit Constant Offset Indexed Addressing
  - The base index register can be X, Y, SP, or PC.
  - This mode allows access any location in the 64-KB range.
  - Examples
    - ldaa 2000,X
    - staa 4000,Y
- 16-bit Constant Indirect Indexed Addressing
  - A 16-bit offset is added to the base index register to form the address of a memory location that contains a pointer to the memory location affected by the instruction.
  - The square brackets distinguish this addressing mode from the 16-bit constant offset indexing.
    - Example,
      - ldaa [10,X]
      - staa [20,Y]

Auto Pre/Post Decrement/Increment Indexed Addressing

- The base index register can be X, Y, or SP.
- The index register can be incremented or decremented by an integer value either before or after indexing taking place.
- The index register retains the changed value after indexing.
- The value to be incremented or decremented is in the ranges -8 thru -1 or 1 thru 8.
- The value needs to be related to the size of the operand or the current instruction.
- Examples
  - staa 1,-SP
  - staa 1,SP-
  - ldx 2,+SP
  - ldx 2,SP+

Accumulator Offset Indexed Addressing

- The effective address of the operand is the sum of the accumulator and the base index register.
  - The base register can be X, Y, SP, or PC.
  - The accumulator can be the 8-bit A or B or the 16-bit accumulator D.
  - Example
    - ldaa B,X
    - stab B,Y

Accumulator D Indirect Indexed Addressing

The value in D is added to the value in the base index register to form the address of the memory location that contains the address to the memory location affected by the instruction.

The square brackets distinguish this addressing mode from accumulator D offset indexing.

Example

```
            jmp     [D,PC]
go1         dc.w    target1
go2         dc.w    target2
go3         dc.w    target3
            …
target1     …
            .
target2     …
            .
target3     …
```