HCS12 Instructions

A Sample of HCS12 Instructions

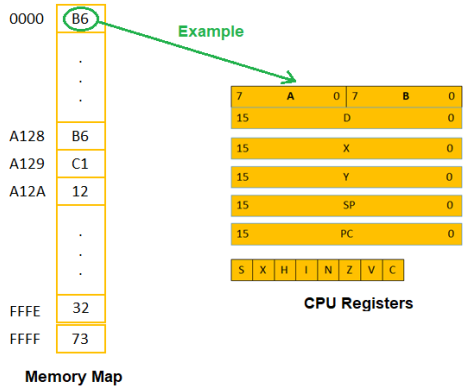
- It would be very helpful to learn a small set of HCS12 instructions that are used most often before we formally learn HCS12 assembly language programming.

- We will examine data movement, addition, and subtraction instructions.

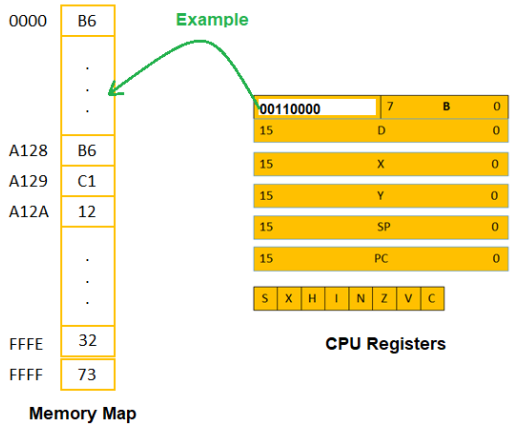
- The HCS12 provides a large group of data movement instructions.
  - Some of them may transfer data between a CPU register and a memory location.
  - Some of them may transfer or exchange data between two registers.
  - Others may transfer data from one memory location to another memory location.

Load and Store Instructions

- Load Instruction:
  - Copies the contents of a memory location or places an immediate value into an accumulator or a register.
  - Memory contents are not changed.



- Store Instruction:
  - Copies the contents of a CPU register into a memory location.
  - The contents of the accumulator or CPU register are not changed.
  - Store instructions automatically update the N and Z flags in the condition code register (CCR).



Load and Store Instructions

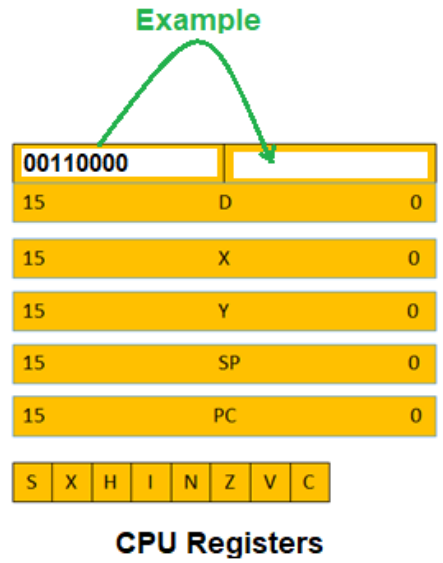
- Example 1:
  - ldaa 0,X
  - **Loads** the contents of the memory location pointed to by index register X into accumulator A
- Example 2:
  - ldab $1004
  - **Loads** the contents of the memory location at $1004 into accumulator B
- Example 3:
  - staa $20
  - **Stores** the contents of accumulator A in the memory location at $20
- Example 4:
  - stx $8000
  - **Stores** the contents of index register X in memory locations at $8000 and $8001

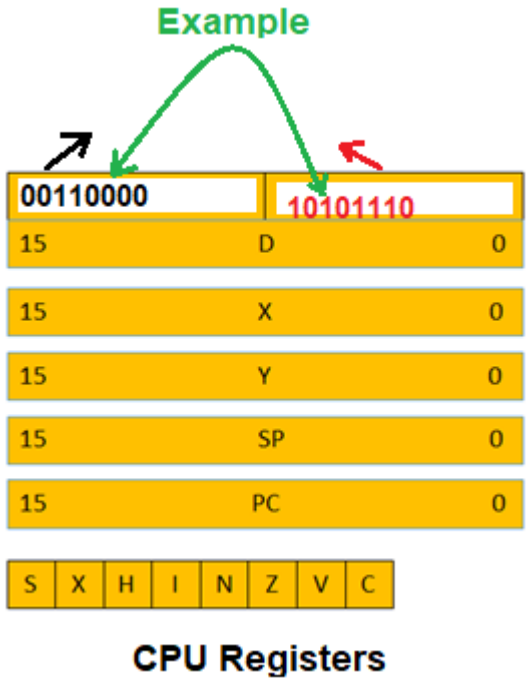
Load Instructions		
Mnemonic	Function	Operation
LDAA <opr>	Load A	A ← [opr]
LDAB <opr>	Load B	B ← [opr]
LDD <opr>	Load D	A:B ← [opr]:[opr+1]
LDS <opr>	Load SP	SP ← [opr]:[opr+1]
LDX <opr>	Load index register X	X ← [opr]:[opr+1]
LDY <opr>	Load index register Y	Y ← [opr]:[opr+1]
LEAS <opr>	Load effective address into SP	SP ← effective address
LEAX <opr>	Load effective address into X	X ← effective address
LEAY <opr>	Load efective address into Y	Y ← effective address
Store Instructions		
Mnemonic	Function	Operation
STAA <opr>	Store A in a memory location	m[opr] ← [A]
STAB <opr>	Store B in a memory location	m[opr] ← [B]
STD <opr>	Store D in a memory location	m[opr]:m[opr+1] ← [A]:[B]
STS <opr>	Store SP in a memory location	m[opr]:m[opr+1] ← [SP]
STX <opr>	Store X in a memory location	m[opr]:m[opr+1] ← [X]
STY <opr>	Store Y in a memory location	m[opr]:m[opr+1] ← [Y]

Transfer and Exchange Instructions

- Transfer instructions:
  - Copy the contents of a register or accumulator into another register or accumulator.
  - Source content is not changed by the operation.

- Exchange instructions:
  - Exchange the contents of pairs of registers or accumulators.





Transfer and Exchange Instructions

- Transfer Instruction:
  - It is possible to transfer from a smaller register to a larger one or vice versa.
  - When transferring from a smaller register to a larger one, the smaller register is signed-extended to 16-bit and then assigned to the larger register.
  - Example 1:
    - **tfr A,X ; A is signed-extended to 16 bits and then assigned to X**
  - When transferring from a larger register to a smaller one, the smaller register receives the value of the lower half of the larger register.
  - Example 2:
    - **tfr X,B ; BI X[7:0], B receives bits 7 to 0 of X**

- Exchange Instruction:
  - Example 1:
    - **exg A, B ;Exchanges the contents of accumulator A and B.**
    - **exg D,X ;Exchanges the contents of double accumulator D and index register X.**

Transfer Instructions		
Mnemonic	Function	Operation
TAB	Transfer A to B	B ← [A]
TAP	Transfer A to CCR	CCR ← [A]
TBA	Transfer B to A	A ← [B]
TFR	Transfer register to register	A, B, CCR, D, X, Y, or SP ← [A, B, CCR, D, X, Y, or SP]
TPA	Transfer CCR to A	A ← [CCR]
TSX	Transfer SP to X	X ← [SP]
TSY	Transfer SP to Y	Y ← [SP]
TXS	Transfer X to SP	SP ← [X]
TYS	Transfer Y to SP	SP ← [Y]
Exchange Instructions		
Mnemonic	Function	Operation
EXG	Exchange register to register	[A, B, CCR, D, X, Y, or SP] ⇔ [A, B, CCR, D, X, Y, or SP]
XGDX	Exchange D with X	[D] ⇔ [X]
XGDY	Exchange D with Y	[D] ⇔ [Y]
Sign Extension Instructions		
Mnemonic	Function	Operation
SEX	Sign extend 8-bit operand	X, Y, or SP ← [A, B, CCR]

Move Instructions

- These instructions move data bytes or words from a source to a destination in memory.
- Six combinations of immediate, extended, and indexed **addressing** are allowed to specify source and destination addresses as shown:
  - IMM ➡ EXT,
  - IMM ➡ DX,
  - EXT ➡ EXT,
  - EXT ➡ IDX,
  - IDX ➡ EXT,
  - IDX ➡ IDX.

Transfer Instructions		
Mnemonic	Function	Operation
MOVB <src>, <dest> MOVW <src>, <dest>	Move byte (8-bit) Move word (16-bit)	dest ← [src] dest ← [src]

- Move instructions allow the user to transfer data from **memory to memory** or from **I/O registers to memory** and vice versa.
- Example:
  - **movb $1000, $2000** **;copies the contents of the memory location at $1000 to the memory location at $2000**
  - **movw 0,X, 0,Y** **;copies the 16-bit word pointed to by X to the memory location pointed to by Y**

Add and Subtract Instructions

- Add and subtract instructions allow the HCS12 to perform fundamental arithmetic operations.

Add Instructions		
Mnemonic	Function	Operation
ABA	Add B to A	A ← [A] + [B]
ABX	Add B to X	X ← [X] + [B]
ABY	Add B to Y	Y ← [Y] + [B]
ADCA <opr>	Add with carry to A	A ← [A] + [opr] + C
ADCB <opr>	Add with carry to B	B ← [B] + [opr] + C
ADDA <opr>	Add without carry to A	A ← [A] + [opr]
ADDB <opr>	Add without carry to B	B ← [B] + [opr]
ADDD <opr>	Add without carry to D	D ← [D] + [opr]
Subtract Instructions		
Mnemonic	Function	Operation
SBA	Subtract B from A	A ← [A] – [B]
SBCA <opr>	Subtract with borrow from A	A ← [A] – [opr] – C
SBCB	Subtract with borrow from B	B ← [B] – [opr] – C
SUBA <opr>	Subtract memory from A	A ← [A] – [opr]
SUBB <opr>	Subtract memory from B	B ← [B] – [opr]
SUBD <opr>	Subtract memory from D	D ← [D] – [opr]

Example 1:

- Write an instruction sequence to add 3 to the memory locations at $10 and $15.

Example1:

- **Question**: Write an instruction sequence to add 3 to the memory locations at $10 and $15.

- **Solution**:
  - A memory location cannot be the destination of an ADD instruction.
  - Therefore, we need to copy the memory content into an accumulator, add 3 to it, and then store the sum back to the same memory location.

```
ldaa    $10     ; copy the contents of memory location at $10 to A
adda    #3      ; add 3 to A
staa    $10     ; store the sum back to memory location at $10
ldaa    $15     ; copy the contents of memory location at $15 to A
adda    #3      ; add 3 to A
staa    $15     ; store the sum back to memory location at $15
```

Example 2:

- Write an instruction sequence to add the byte pointed to by index register X and the following byte and place the sum at the memory location pointed to by index register Y.

Example 2:

- **Question:** Write an instruction sequence to add the byte pointed to by index register X and the following byte and place the sum at the memory location pointed to by index register Y.

- **Solution:**
  - The byte pointed to by index register X and the following byte can be accessed by using the indexed addressing mode.

```
ldaa    0,X     ; put the byte pointed to by X in A
adda    1,X     ; add the following byte to A
staa    0,Y     ; store the sum at the location pointed to by Y
```

Example 3:

- Write an instruction sequence to add the numbers stored at $1000 and $1001 and store the sum at $1004.

Example 3:

- **Question:** Write an instruction sequence to add the numbers stored at $1000 and $1001 and store the sum at $1004.

- **Solution:**
  - To add these two numbers, we need to put one of them in an accumulator.

```
ldaa    $1000    ; copy the number stored in memory location at $1000 to A
adda    $1001    ; add the second number to A
staa    $1004    ; save the sum at memory location at $1004
```

Example 4:

- Write an instruction sequence to swap the 2 bytes at $100 and $200.

Example 4:

- **Question:** Write an instruction sequence to swap the 2 bytes at $100 and $200.

- **Solution:**
  - To swap the 2 bytes, we need to make a copy of one of the 2 bytes and then the swapping can proceed.

```
ldaa    $100        ; make a copy of m[$100] in A
movb    $200,$100   ; store [$200] in m[$100]
staa    $200        ; store the original [$100] in m[$200]
```