Microcomputers I – CE 320

Mohammad Ghamari, Ph.D.
Electrical and Computer Engineering
Kettering University

Announcements

- Your first quiz is next Thursday!
  - Topic: Introduction to Microcomputers and Number Systems
  - Study: Homework Exercise 1 and class lectures.

Lecture 4: Addressing Modes

Today’s Goals

Two major goals

- Understand addressing modes so figure out how to use them.
  - If you don’t get addressing modes, you will have a serious problem to complete this course.
- Learn how to use a program trace.

Addressing Modes

How to get effective addresses

- The operand of an instruction can use different methods for **specifying data in the memory** (=addressing modes).
  - If the data number is in registers (inside the microprocessor), a memory address is not needed.
  - The addressing mode may specify a value, a register, or a memory location to be used as an operand.

- The HCS12 has six addressing modes
  - Extended (EXT)
  - Direct (DIR)
  - Inherent (INH)
  - Immediate (IMM)
  - Index (IDX)
  - Relative (REL) : Used only with branch instructions.

Remember that Instruction codes consist of **Op code** and **Operand.**

- Effective Address
  - The effective address is the location that holds the data to be used by the operation.
  - The operand is often used to construct the effective address.
  - An addressing mode tells the microprocessor the way of calculation to get the effective address.

- 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.

Addressing Modes

Methods for specifying a particular address in memory

- **1. Extended –** 16-bit absolute address in the instruction.
- **2. Direct –** 8-bit absolute address is in the instruction.
- **3. Inherent –** not really an addressing mode, there is no memory address specified.
- **4. Immediate –** Data itself is part of the instruction.
- **5. Relative –** Offset relative to the instruction itself specifies a branch target address.
- **6. Indexed –** A base address + offset point to the data.
  - **Indexed-indirect –** A base address + offset point to an address, which points to the data.

Extended Addressing (EXT)

Also called Absolute Addressing

-The most straightforward mode

HCS12 Instruction: **LDAA** **Load Accumulator A**

Operation: (M) ⇒ A

Description: Loads the content of memory location M into accumulator A. The condition codes are set according to the data.

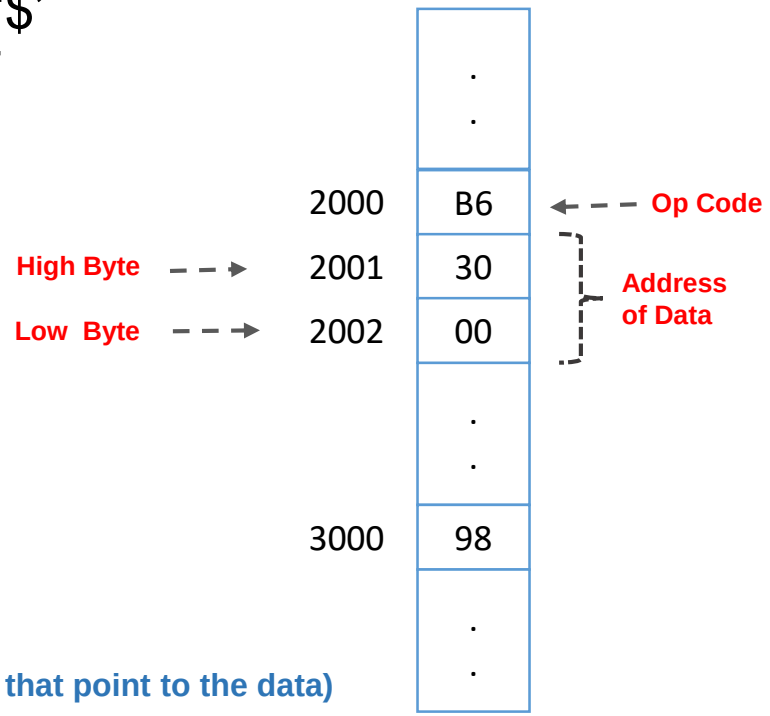
Source Form	Address Mode	Object Code
LDAA *#opr8i*	IMM	86 ii
LDAA *opr8a*	DIR	96 dd
LDAA *opr16a*	EXT	B6 hh ll
LDAA *oprx0_xysp*	IDX	A6 xb
LDAA *oprx9,xysp*	IDX1	A6 xb ff
LDAA *oprx16,xysp*	IDX2	A6 xb ee ff
LDAA [D,*xysp*]	[D,IDX]	A6 xb
LDAA [*oprx16,xysp*]	[IDX2]	A6 xb ee ff

Table shows different machine codings (object codes) for the available addressing modes

- Effective address: Is specified in two bytes following the op code
  - No operation needed.
  - Extended addressing tells the full memory address.

- Format:
  - Two-byte hexadecimal number (4-digit) preceded with a $. Actually '$' simply means that the number is a hexadecimal number. (A number could be followed by 'h' excluding '').

- Example:
  - (Assuming the instruction is stored at $2000)
  - LDAA $3000
    - Load a byte value stored at address $3000 into the register A.
    - LDAA *opr16a* (M) ➔ A EXT B6 hh ll
    - 98 ➔ A

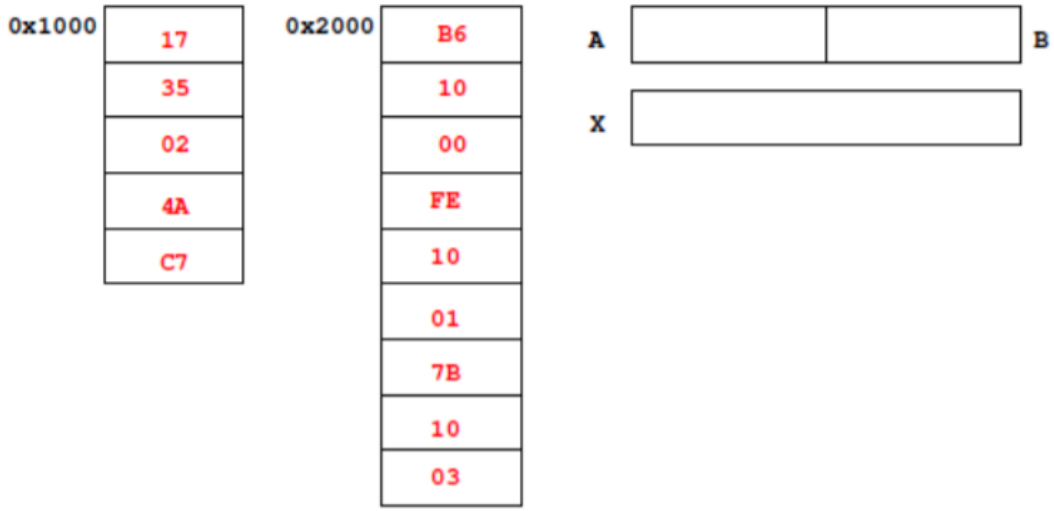


Extended Addressing (EXT)

Example 1:

LDAA $1000 ; ($1000) → A

1. What is the effective address?
2. How the effective address is specified?



HCS12 Instruction: **LDAA** **Load Accumulator A**

Operation: (M) ⇒ A

Description: Loads the content of memory location M into accumulator A. The condition codes are set according to the data.

Source Form	Address Mode	Object Code
LDAA *#opr8i*	IMM	86 ii
LDAA *opr8a*	DIR	96 dd
LDAA *opr16a*	EXT	B6 hh ll
LDAA *oprx0_xysp*	IDX	A6 xb
LDAA *oprx9,xysp*	IDX1	A6 xb ff
LDAA *oprx16,xysp*	IDX2	A6 xb ee ff
LDAA [D,*xysp*]	[D,IDX]	A6 xb
LDAA [*oprx16,xysp*]	[IDX2]	A6 xb ee ff

Remember:

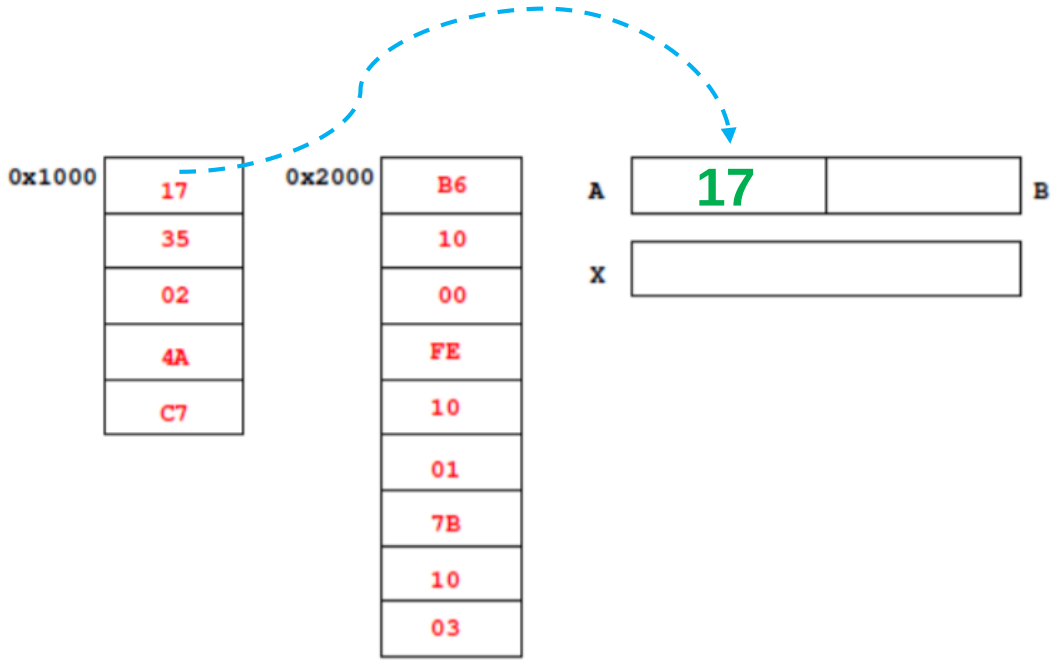
- **Effective address** is the **location (address)** that holds the **data** to be used by the operation.
- Effective address is specified in two bytes following the op code.

Extended Addressing (EXT)

Example 1 (Solution):

LDAA $1000 ; ($1000) → A

1. What is the effective address? **$1000**
2. How the effective address is specified? **B6 10 00**



HCS12 Instruction: **LDAA** **Load Accumulator A**

Operation: (M) ⇒ A

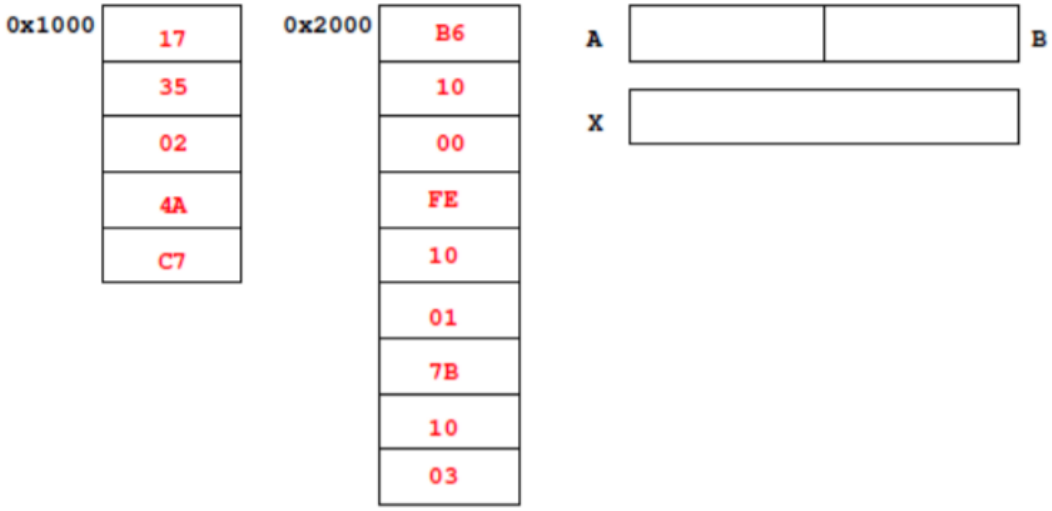
Description: Loads the content of memory location M into accumulator A. The condition codes are set according to the data.

Source Form	Address Mode	Object Code
LDAA *#opr8i*	IMM	86 ii
LDAA *opr8a*	DIR	96 dd
LDAA *opr16a*	EXT	B6 hh ll
LDAA *oprx0_xysp*	IDX	A6 xb
LDAA *oprx9,xysp*	IDX1	A6 xb ff
LDAA *oprx16,xysp*	IDX2	A6 xb ee ff
LDAA [D,*xysp*]	[D,IDX]	A6 xb
LDAA [*oprx16,xysp*]	[IDX2]	A6 xb ee ff

Extended Addressing (EXT)

Example 2:

LDX $1001 ; ($1001:$1002) → X



1. What is the effective address?
2. How the effective address is specified?

HCS12 Instruction: **LDX** **Load Accumulator X**

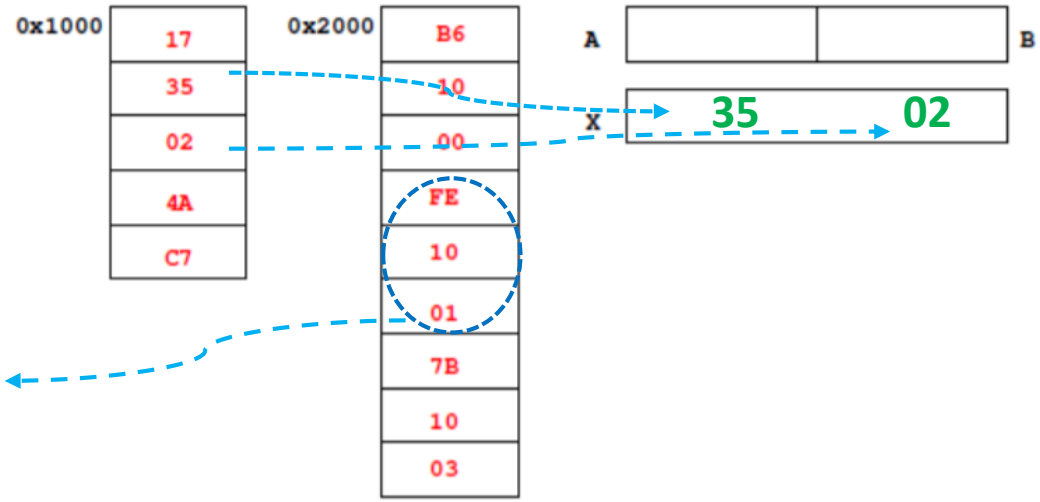
Description: **Load the content of memory location M and M+1 into double accumulator X**

LDX #*opr16i*	(M:M+1) ⇒ X	IMM	CE jj kk
LDX *opr8a*	Load Index Register X	DIR	DE dd
LDX *opr16a*		EXT	FE hh ll
LDX *oprx0_xysp*		IDX	EE xb
LDX *oprx9,xysp*		IDX1	EE xb ff
LDX *oprx16,xysp*		IDX2	EE xb ee ff
LDX [D,*xysp*]		[D,IDX]	EE xb
LDX [*oprx16,xysp*]		[IDX2]	EE xb ee ff

Extended Addressing (EXT)

Example 2 (Solution):

LDX $1001 ; ($1001:$1002) → X



1. What is the effective address? **$1001**
2. How the effective address is specified? **FE 10 01**

HCS12 Instruction: **LDX** **Load Accumulator X**

Description: **Load the content of memory location M and M+1 into double accumulator X with a 16-bit from value from memory**

LDX *#opr16i*	(M:M+1) ⇒ X	IMM	CE jj kk
LDX *opr8a*	Load Index Register X	DIR	DE dd
LDX *opr16a*		EXT	FE hh ll
LDX *oprx0_xysp*		IDX	EE xb
LDX *oprx9,xysp*		IDX1	EE xb ff
LDX *oprx16,xysp*		IDX2	EE xb ee ff
LDX [D,*xysp*]		[D,IDX]	EE xb
LDX [*oprx16,xysp*]		[IDX2]	EE xb ee ff

Direct Addressing (DIR)

Also called Zero-Paging Addressing

- Effective address:
  - This addressing mode only supplies the lower byte of the address.
  - Extend the one byte address to two-bytes by concatenating $00 to the beginning of the operand.
- Format:
  - One byte hexadecimal number (2-digit) preceded with a $.
- Example:
  - (Assuming the instruction is stored at $2000)
  - LDAA $80
    - Load a byte value stored at address $0080 into the register A.
    - LDAA *opr8a* (M) ➔ A DIR 96 dd
      - 98 ➔ A

Op Code

One byte of operand (an address that point to the data)

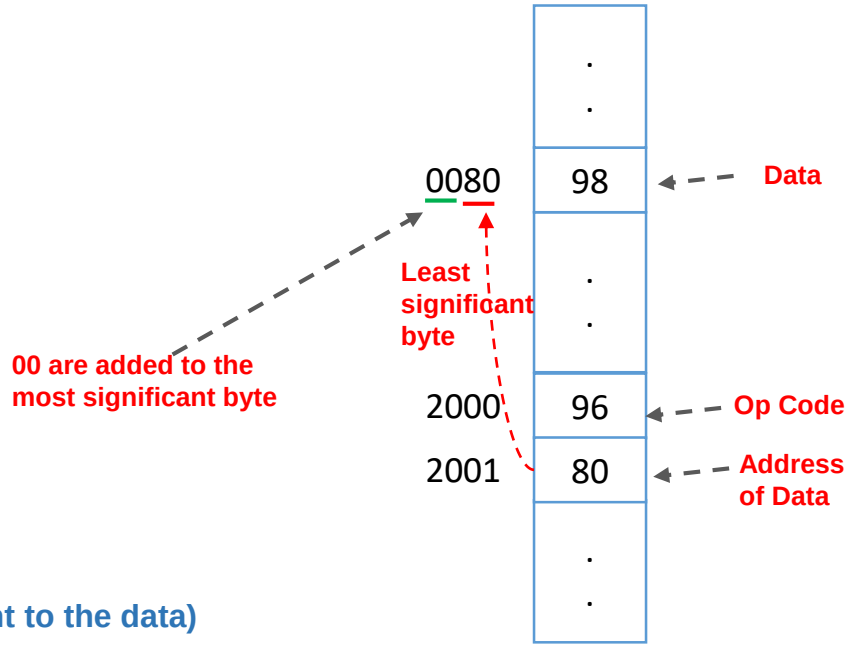
HCS12 Instruction: **LDAA** **Load Accumulator A**

Operation: (M) ⇒ A

Description: Loads the content of memory location M into accumulator A. The condition codes are set according to the data.

Source Form	Address Mode	Object Code
LDAA *#opr8i*	IMM	86 ii
LDAA *opr8a*	DIR	96 dd
LDAA *opr16a*	EXT	B6 hh ll
LDAA *oprx0_xysp*	IDX	A6 xb
LDAA *oprx9,xysp*	IDX1	A6 xb ff
LDAA *oprx16,xysp*	IDX2	A6 xb ee ff
LDAA [D,*xysp*]	[D,IDX]	A6 xb
LDAA [*oprx16,xysp*]	[IDX2]	A6 xb ee ff

Table shows different machine codings (object codes) for the available addressing modes



Direct Addressing (DIR)

Also called Zero-Paging Addressing

- Can be used to address only 256 different locations, only one byte would be needed to specify an individual location.
  - Have to decide in advance which 256-byte **page** in memory the one-byte address refers to.
  - If choose the zero'th one (at addresses 0000–00FF), you have the direct addressing, or **zero-page addressing**, mode.

Notes:

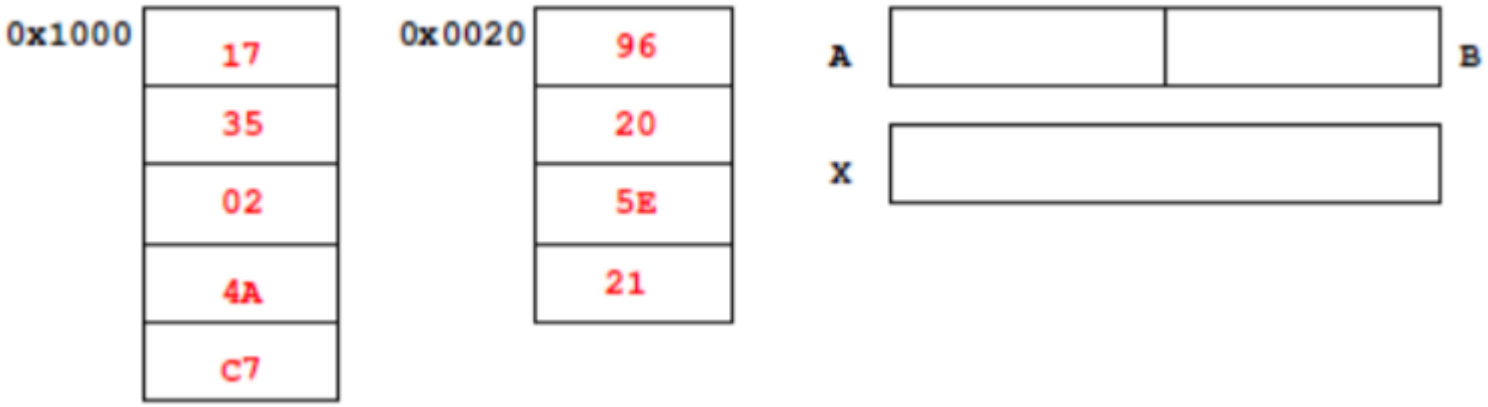
- Takes one byte, i.e., less memory space and execution time than extended addressing.
- Only a small range (0000–00FF) of memory locations can be addressed.

Direct Addressing (DIR)

Example 1:

LDAA $20 ; ($0020) → A

1. What is the effective address?
2. How the effective address is specified?
3. What is the object code?



HCS12 Instruction: **LDAA** **Load Accumulator A**

Operation: (M) ⇒ A

Description: Loads the content of memory location M into accumulator A. The condition codes are set according to the data.

Source Form	Address Mode	Object Code
LDAA *#opr8i*	IMM	86 ii
LDAA *opr8a*	DIR	96 dd
LDAA *opr16a*	EXT	B6 hh ll
LDAA *oprx0_xysp*	IDX	A6 xb
LDAA *oprx9,xysp*	IDX1	A6 xb ff
LDAA *oprx16,xysp*	IDX2	A6 xb ee ff
LDAA [D,*xysp*]	[D,IDX]	A6 xb
LDAA [*oprx16,xysp*]	[IDX2]	A6 xb ee ff

Direct Addressing (DIR)

Example 1 (Solution):

LDAA $20 ; ($0020) → A

1. What is the effective address? **$0020**
2. How the effective address is specified?
3. What is the object code? **96 20**

HCS12 Instruction: **LDAA** **Load Accumulator A**

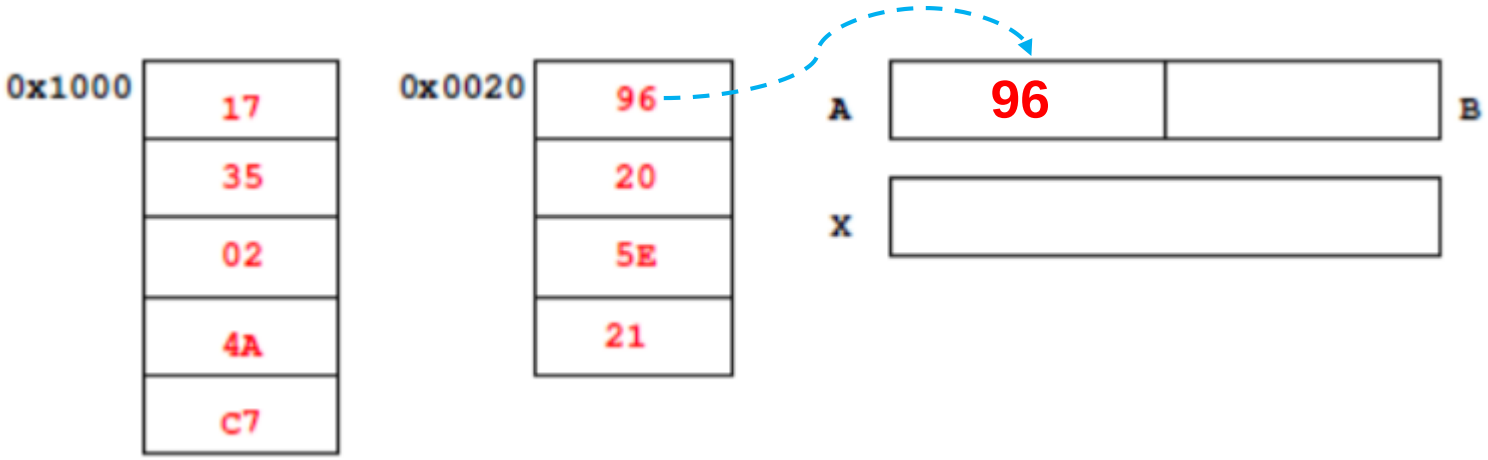
Operation: (M) ⇒ A

Description: Loads the content of memory location M into accumulator A. The condition codes are set according to the data.

Source Form	Address Mode	Object Code
LDAA *#opr8i*	IMM	86 ii
LDAA *opr8a*	DIR	96 dd
LDAA *opr16a*	EXT	B6 hh ll
LDAA *oprx0_xysp*	IDX	A6 xb
LDAA *oprx9,xysp*	IDX1	A6 xb ff
LDAA *oprx16,xysp*	IDX2	A6 xb ee ff
LDAA [D,*xysp*]	[D,IDX]	A6 xb
LDAA [*oprx16,xysp*]	[IDX2]	A6 xb ee ff

Remember:

- **Effective address** is the **location** (**address**) that holds the **data** to be used by the operation.
- Effective address is specified using the **one byte** following the **op code** as the least significant byte and **00** as the most significant.



Inherent Addressing (INH)

Also called Implied Addressing

- Instructions operate on data in registers only, not stored in memory somewhere.
  - **Only an op code is required**.

- Effective address:
  - No “effective address” in memory; only data in registers is used.
- Format:
  - Inherent instructions **have no operands**.

- Example:
  - INCA
    - Increase register A by 1
    - INCA (A) + $01 ➔ A INH 42
- Note:
  - Takes only one byte; the most compact and fastest instructions.

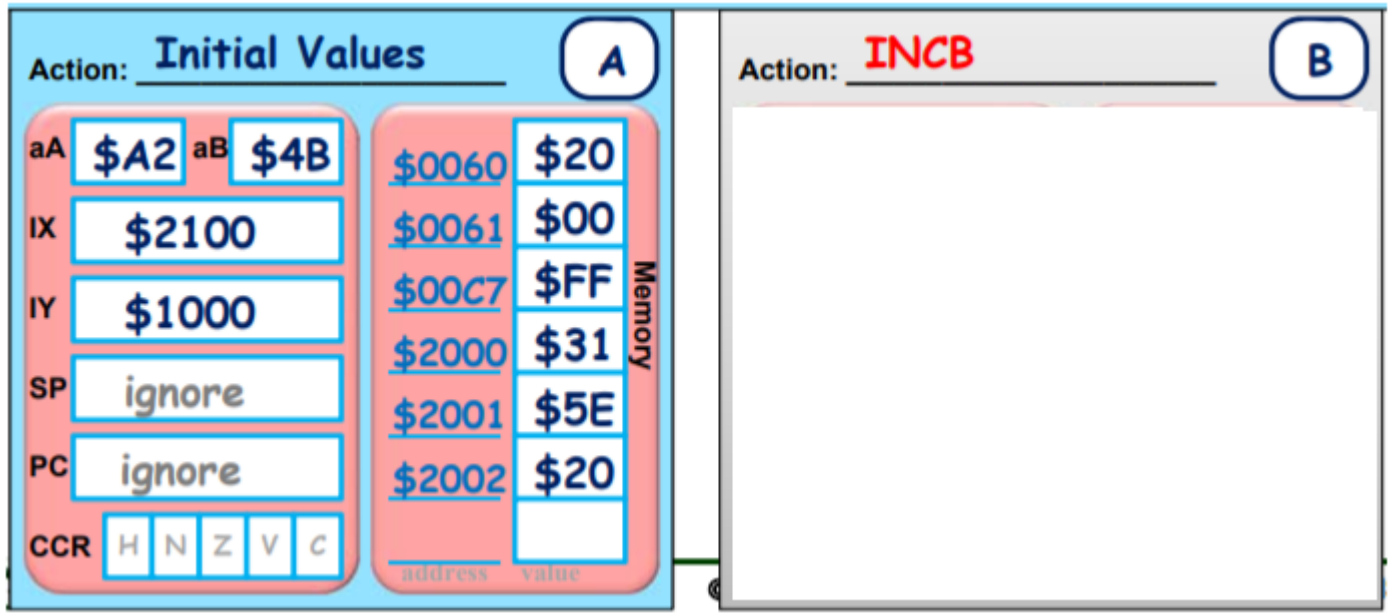
Source Form	Operation		Addr. Mode	Machine Coding (hex)
INCA	(A) + $01 ⇒ A	Increment Acc. A	INH	42
DECA	(A) – $01 ⇒ A	Decrement A	INH	43
DECB	(B) – $01 ⇒ B	Decrement B	INH	53

Inherent Addressing (INH)

Example 1:

Source Form	Operation		Addr. Mode	Machine Coding (hex)
INCB	(B) + $01 ⇒ B	Increment Acc. B	INH	52

INCB

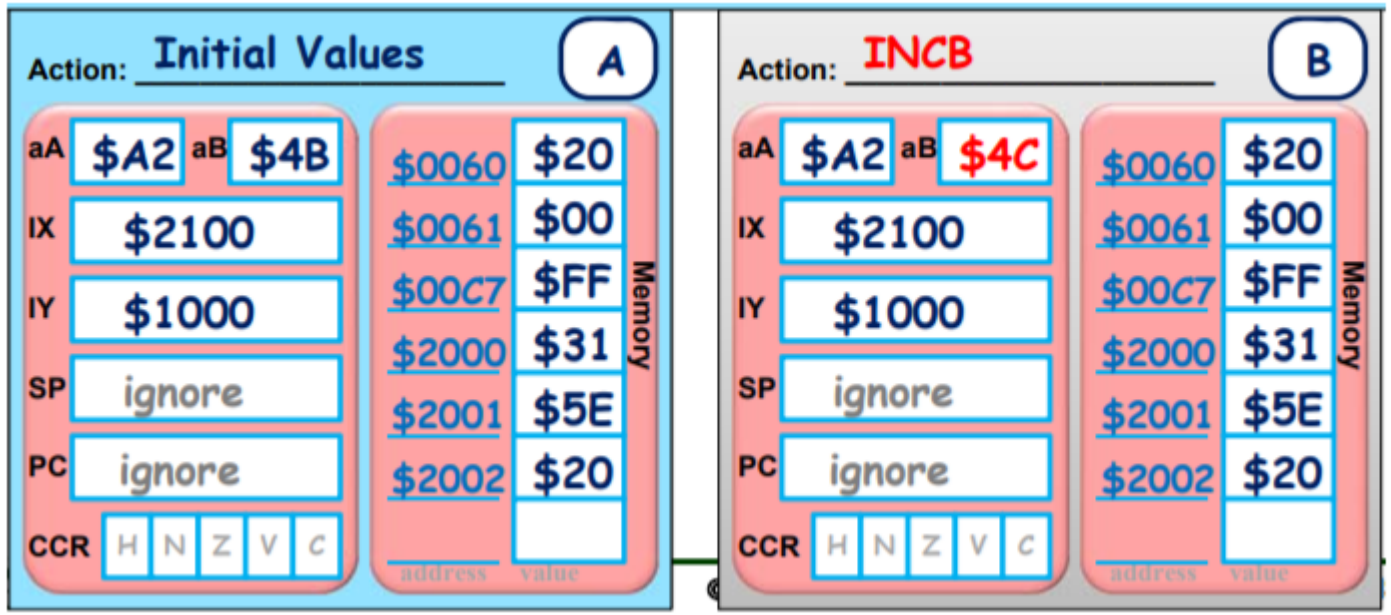


Inherent Addressing (INH)

Example 1 (Solution):

Source Form	Operation		Addr. Mode	Machine Coding (hex)
INCB	(B) + $01 ⇒ B	Increment Acc. B	INH	52

INCB



Inherent

Increment B

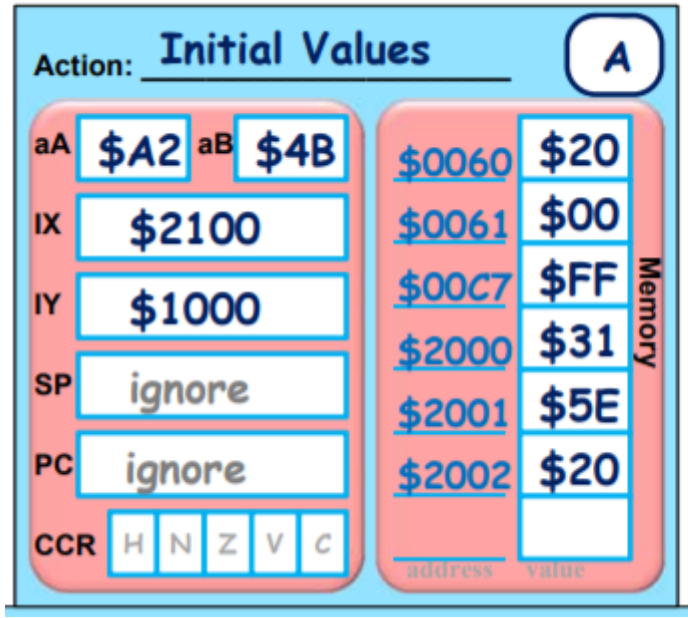
(B ← B+1)

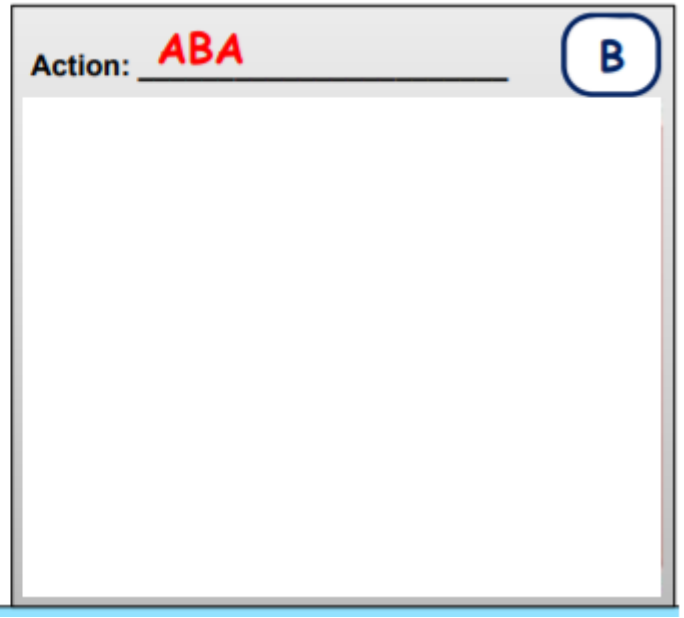
Inherent Addressing (INH)

Example 2:

ABA

Source Form	Operation
ABA	(A) + (B) ⇒ A Add Accumulators A and B



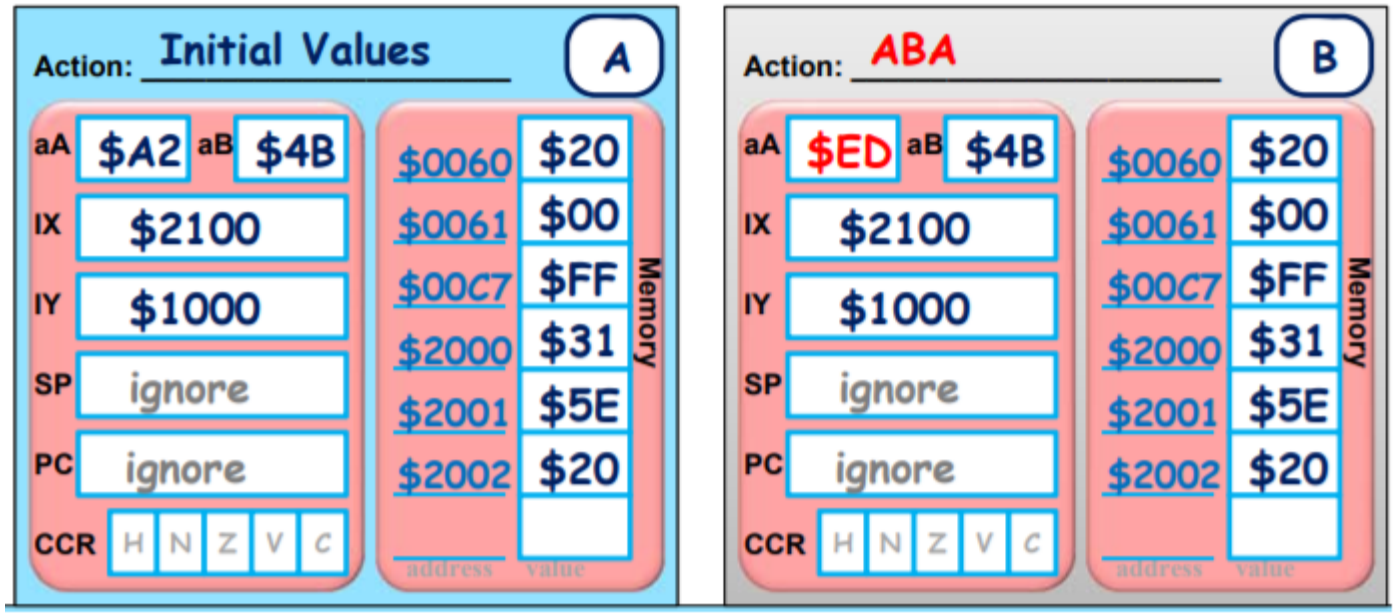


Inherent Addressing (INH)

Example 2 (Solution):

ABA

Source Form	Operation
ABA	(A) + (B) ⇒ A Add Accumulators A and B



Inherent

Add B to A

A2
+4B
=ED

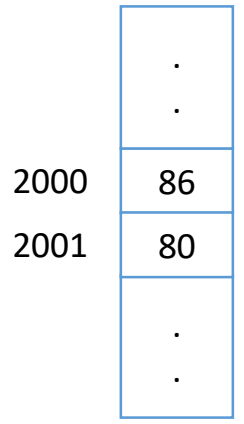
Immediate Addressing (IMM)

- Effective address:
  - No operation. The data itself is supplied as the operand.
- Format:
  - Number preceded with a #. '#' is followed by a number that is a **value** instead of an **address**!
- Example:
  - (Assuming the instruction is stored at $2000)
  - LDAA #$80
    - Load a byte value(the operand itself) into the register A.
    - 80_{16} ➔ A
  - LDD #1000
    - 1000 is $03E8_{16}$ ➔ D (meaning 03 → A and E8 → B)
- The size of an operand
  - Register A and B have **one-byte** immediate operands.
  - Register D, X, Y, SP, and PC have **two-byte** ones.

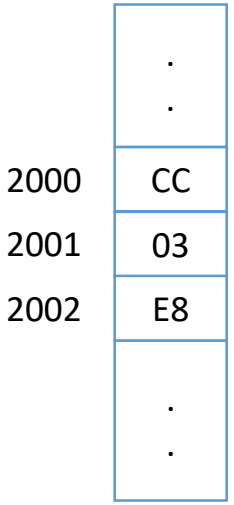
Source Form	Address Mode	Object Code
LDAA *#opr8i*	IMM	86 ii
LDAA *opr8a*	DIR	96 dd
LDAA *opr16a*	EXT	B6 hh ll
LDAA *oprx0_xysp*	IDX	A6 xb
LDAA *oprx9,xysp*	IDX1	A6 xb ff
LDAA *oprx16,xysp*	IDX2	A6 xb ee ff
LDAA [D,*xysp*]	[D,IDX]	A6 xb
LDAA [*oprx16,xysp*]	[IDX2]	A6 xb ee ff

Source Form	Operation	Addr. Mode	Machine Coding (hex)
LDD *#opr16i*	(M:M+1) ⇒ A:B	IMM	CC jj kk
LDD *opr8a*	Load Double Accumulator D (A:B)	DIR	DC dd
LDD *opr16a*		EXT	FC hh ll
LDD *oprx0_xysp*		IDX	EC xb
LDD *oprx9,xysp*		IDX1	EC xb ff
LDD *oprx16,xysp*		IDX2	EC xb ee ff
LDD [D,*xysp*]		[D,IDX]	EC xb
LDD [*oprx16,xysp*]		[IDX2]	EC xb ee ff

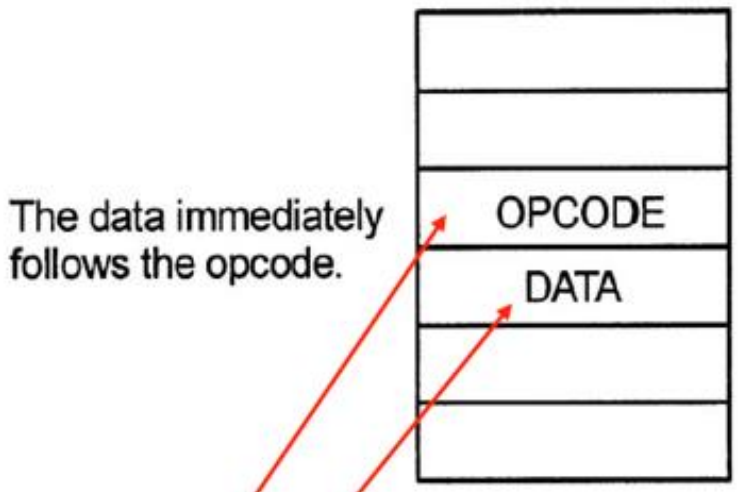
LDAA #$80



LDD #$1000



Immediate Addressing (IMM)



Byte following the opcode is the data itself.

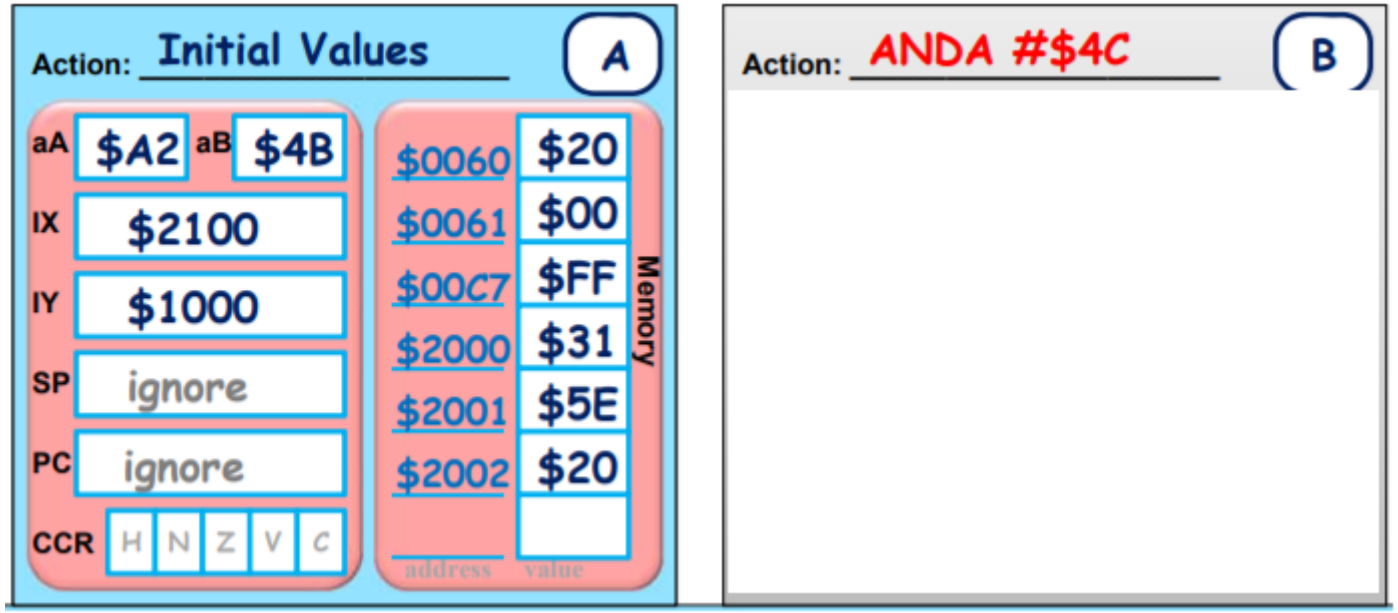
Immediate Addressing Examples

```
21                    ; Immediate Addressing
22
23   8640             ldaa  #64       ; Decimal 64 -> A
24   8664             ldaa  #$64      ; Hexadecimal 64 -> A
25   CE12 34          ldx   #$1234    ; Hexadecimal 1234 -> X
26   180B 4012 34     movb  #64,$1234; Decimal 64 -> $1234
```

Immediate Addressing (IMM)

- Example 1:

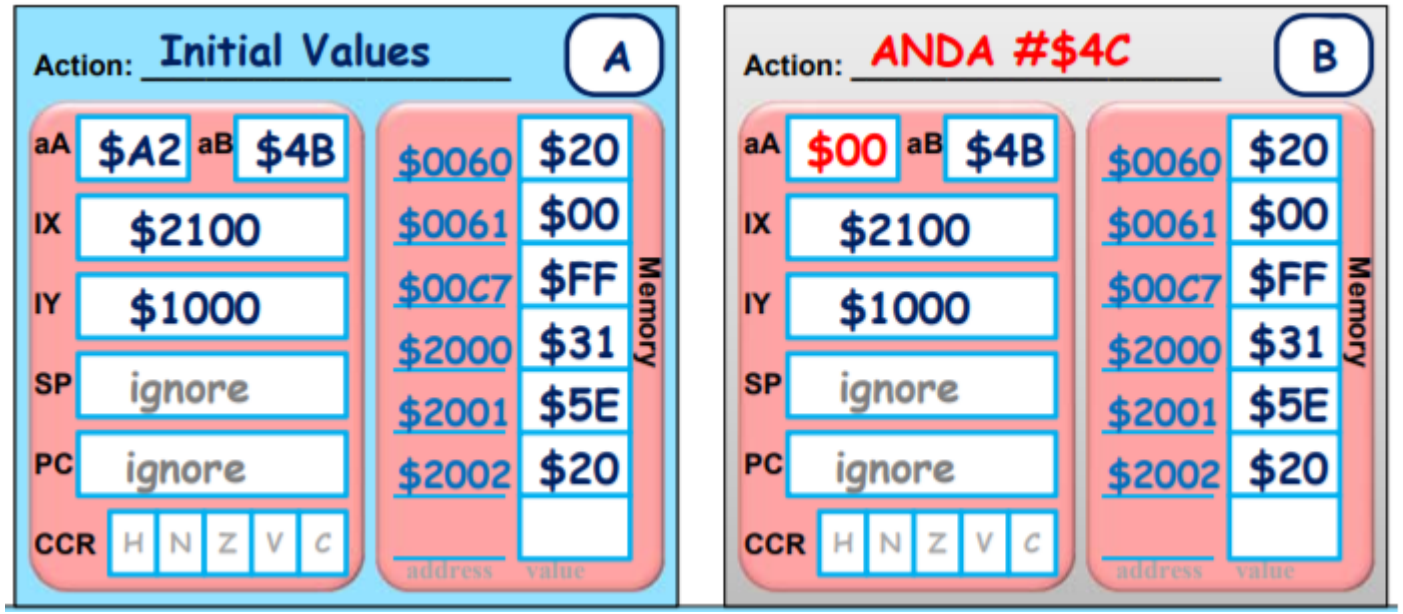
Source Form	Operation	Addr. Mode	Machine Coding (hex)
ANDA *#opr8i*	(A) • (M) ⇒ A	IMM	84 ii
ANDA *opr8a*	Logical AND A with Memory	DIR	94 dd
ANDA *opr16a*		EXT	B4 hh ll
ANDA *oprx0_xysp*		IDX	A4 xb
ANDA *oprx9,xysp*		IDX1	A4 xb ff
ANDA *oprx16,xysp*		IDX2	A4 xb ee ff
ANDA [D,*xysp*]		[D,IDX]	A4 xb
ANDA [*oprx16,xysp*]		[IDX2]	A4 xb ee ff



Immediate Addressing (IMM)

- Example 1 (Solution):

Source Form	Operation	Addr. Mode	Machine Coding (hex)
ANDA *#opr8i*	(A) • (M) ⇒ A	IMM	84 ii
ANDA *opr8a*	Logical AND A with Memory	DIR	94 dd
ANDA *opr16a*		EXT	B4 hh ll
ANDA *oprx0_xysp*		IDX	A4 xb
ANDA *oprx9,xysp*		IDX1	A4 xb ff
ANDA *oprx16,xysp*		IDX2	A4 xb ee ff
ANDA [D,*xysp*]		[D,IDX]	A4 xb
ANDA [*oprx16,xysp*]		[IDX2]	A4 xb ee ff



Immediate

AND aA w/ M

(A2)(4C)

A2 = 1010 0010

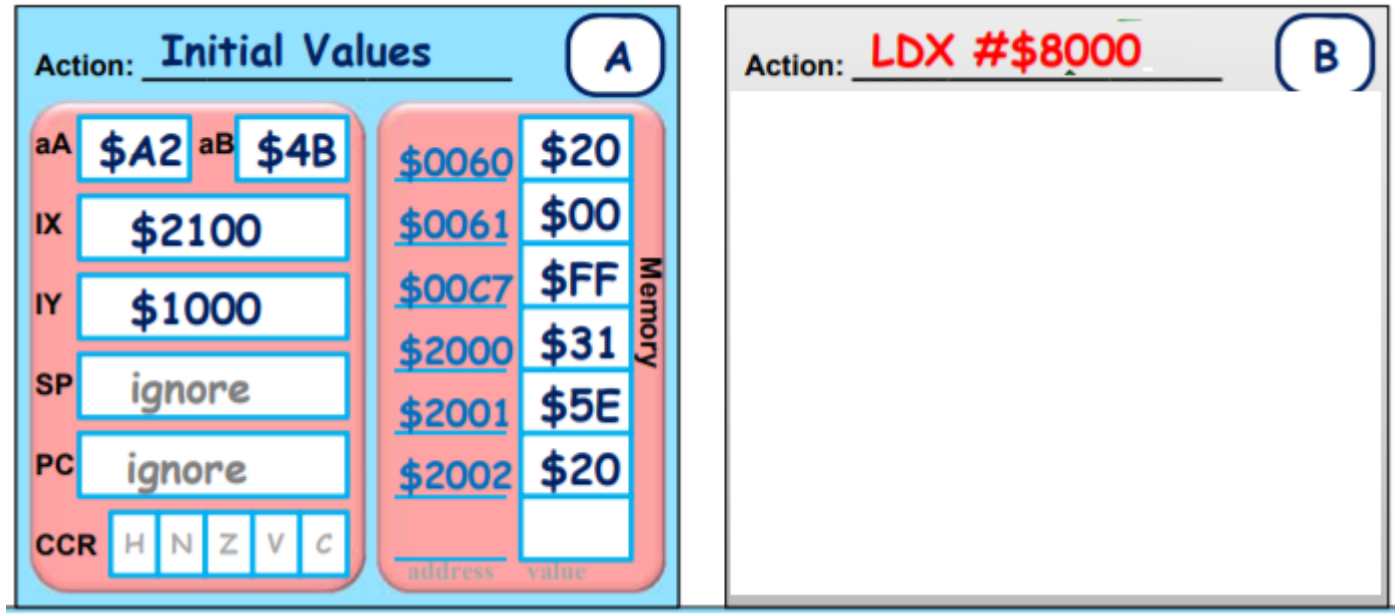
4C = 0100 1101

0000 0000

Immediate Addressing (IMM)

Source Form	Operation	Addr. Mode	Machine Coding (hex)
LDX #*opr16i*	(M:M+1) ⇒ X	IMM	CE jj kk
LDX *opr8a*	Load Index Register X	DIR	DE dd
LDX *opr16a*		EXT	FE hh ll
LDX *oprx0_xysp*		IDX	EE xb
LDX *oprx9,xysp*		IDX1	EE xb ff
LDX *oprx16,xysp*		IDX2	EE xb ee ff
LDX [D,*xysp*]		[D,IDX]	EE xb
LDX [*oprx16,xysp*]		[IDX2]	EE xb ee ff

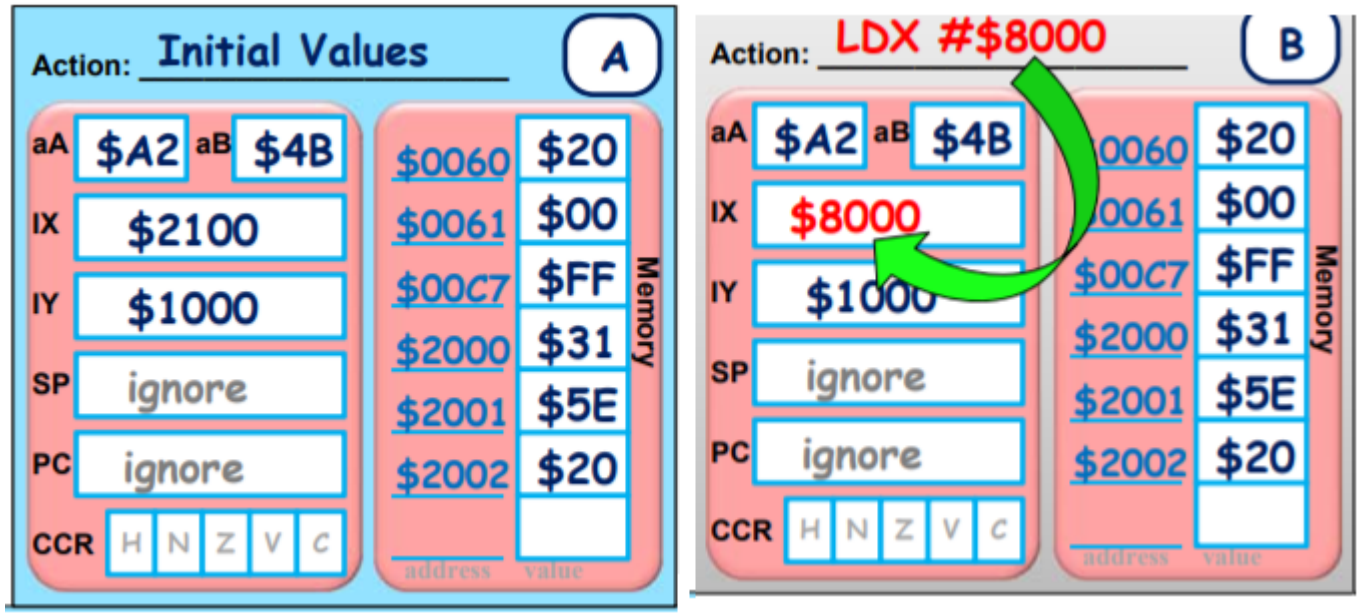
- Example 2:



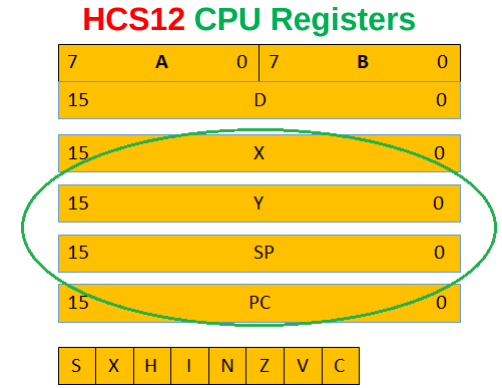
Immediate Addressing (IMM)

Source Form	Operation	Addr. Mode	Machine Coding (hex)
LDX #*opr16i*	(M:M+1) ⇒ X	IMM	CE jj kk
LDX *opr8a*	Load Index Register X	DIR	DE dd
LDX *opr16a*		EXT	FE hh ll
LDX *oprx0_xysp*		IDX	EE xb
LDX *oprx9,xysp*		IDX1	EE xb ff
LDX *oprx16,xysp*		IDX2	EE xb ee ff
LDX [D,*xysp*]		[D,IDX]	EE xb
LDX [*oprx16,xysp*]		[IDX2]	EE xb ee ff

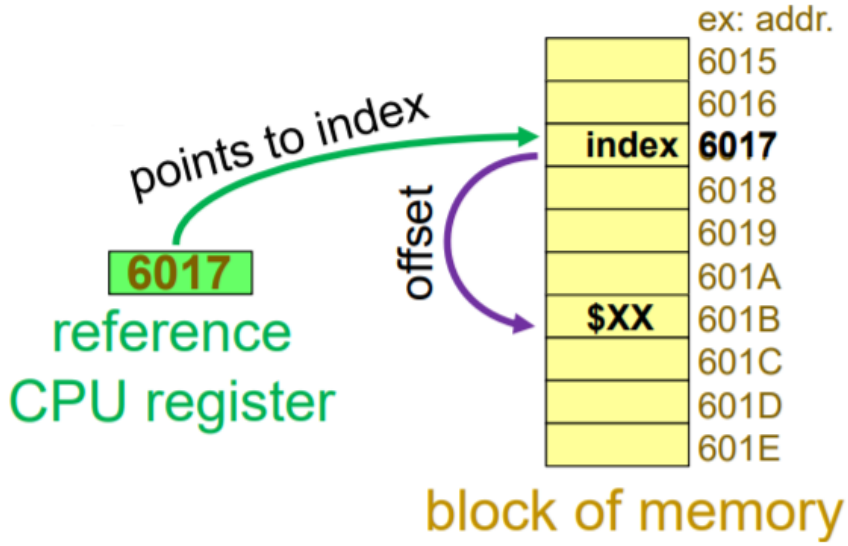
- Example 2 (Solution):



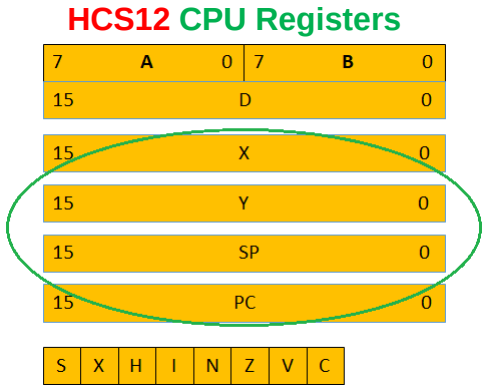
Index Addressing (IDX, IDX1, IDX2)



- Indexed: instruction data is in memory at address specified relative to (offset from) a reference address that is stored in a CPU register.
  - Reference address can be in i**X**, i**Y**, **SP**, or **PC**
  - Offsets are signed number ➡ can offset forward or backward
  - Useful for accessing a list of data beginning (or ending) at the reference address



Index Addressing (IDX, IDX1, IDX2)



- Effective Address
  - Add the operand as a signed number to the value in the X, Y, PC, or S registers.
- Format
  - Signed number, Register (X, Y, PC, or S)
- Example:
  - LDAA 0,X
    - The effective address is the value(=address) in register X. (=X+0)
  - LDD -100,Y
    - The effective address is 100 lower than the value in Y. (=Y-100)
  - LDX 1000, Y
    - The effective address is 1000 higher than the value in Y. (=Y+1000)
- Notes:
  - The value in the specified register is not changed.
  - The smallest number of bits will be used to represent the address.

Index Addressing Postbytes

- An operand in the index addressing is called a **postbyte**.
- The postbyte tells the processor which two-byte register to be used as the base address, and the size of the offset.

Register	rr
X	00
Y	01
SP	10
PC	11

Postbyte for **5-bit** Offset: **rr**0**nnnnn**
Postbytes for **9-bit** Offset: 111**rr**00**n nnnnnnnn**
Postbytes for **16-bit** Offset: 111**rr**010 **nnnnnnnn nnnnnnnn**

Index Addressing

Examples:

-100 (decimal from signed 2's complement) = 9C (hex)

9C (hex) = 10011100 (binary)

Since 9C is represented with 8-bit binary, we cannot use 5-bit offset, thus we use 9-bit offset form.

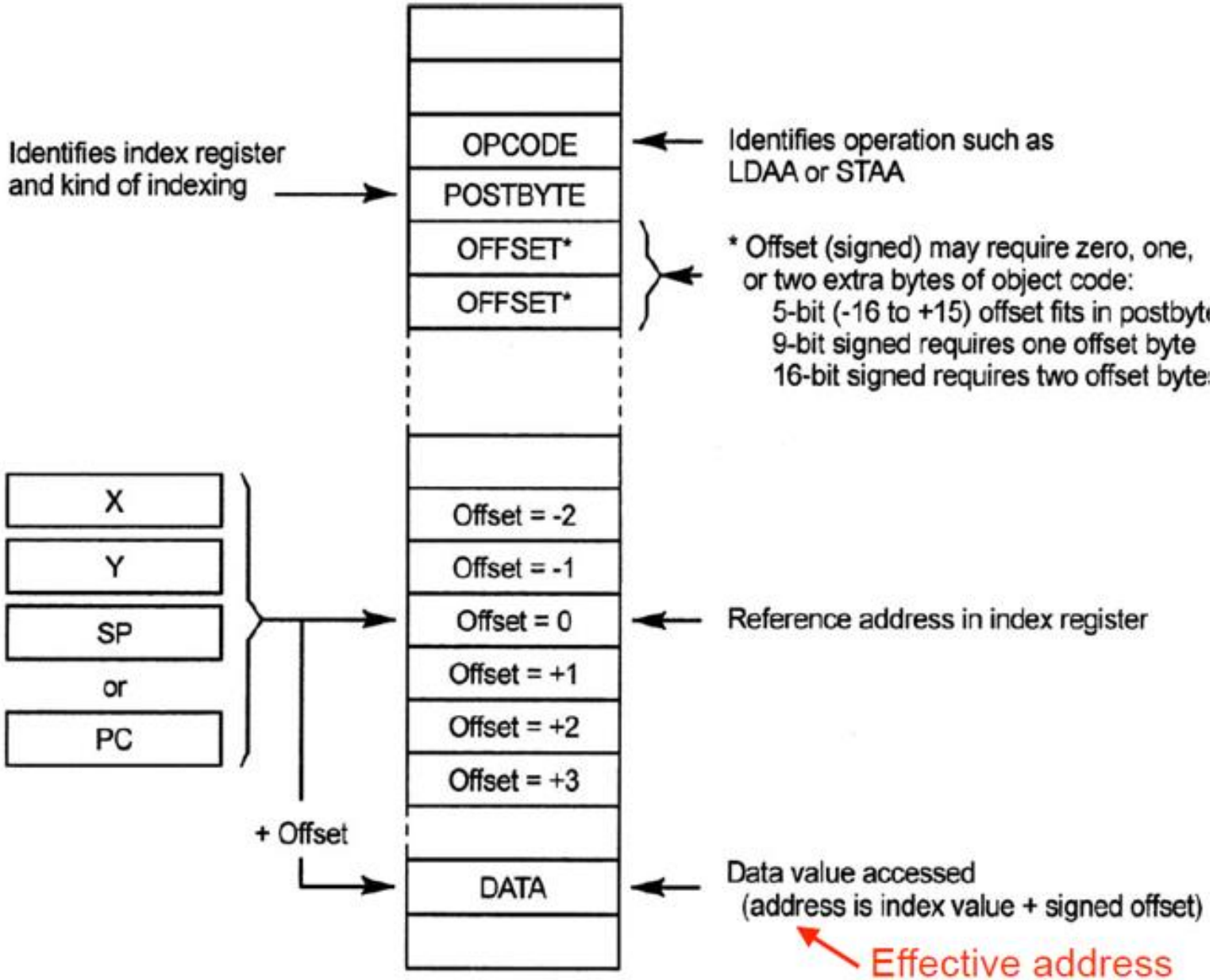
Instruction	Machine Code			
LDAA 4,Y	A6	44		
		01 0 00100		
LDD -100,X	EC	E1	9C	
		111 **00** 00 1	10011100	
LDX -1000,Y	EE	EA	FC	18
		111 **01** 010	1111 1100	0001 1000

4 (decimal) = 4 (hex)
4 (hex) = 0100 (binary)
Since 4 is represented with 4-bit binary and is less than 5-bit offset, we can use 5-bit offset form.

Postbyte for 5-bit Offset: rr0nnnnn
Postbytes for 9-bit Offset: 111rr00n nnnnnnnn
Postbytes for 16-bit Offset: 111rr010 nnnnnnnn nnnnnnnn

Register	rr
X	00
Y	01
SP	10
PC	11

Index Addressing



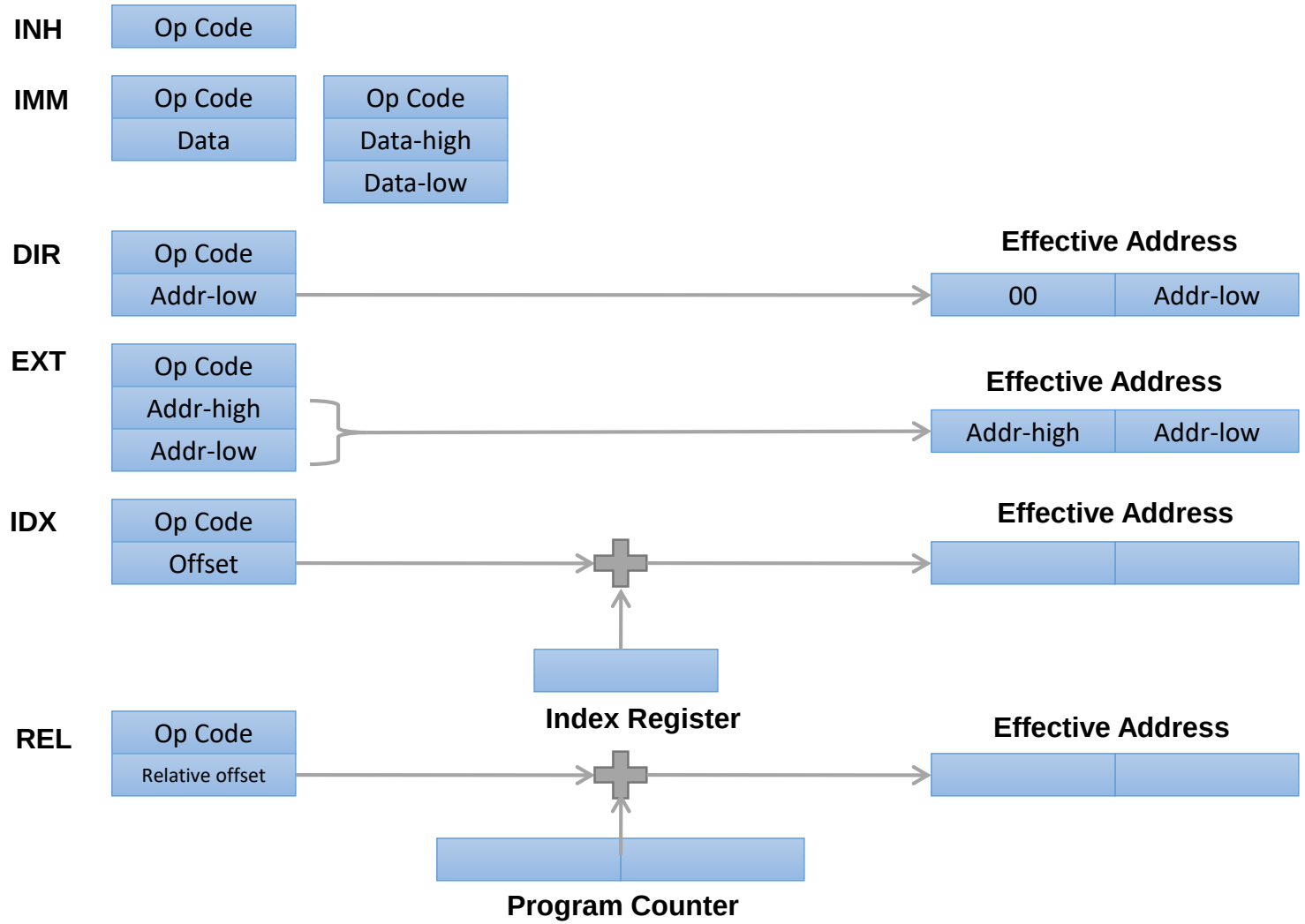
Instruction Set

Source Form	Operation	Addr. Mode	Machine Coding	Access Detail	S X H I	N Z V C
LDAA #opr8i	(M) ⇒ A	IMM	86 ii	P	- - - -	Δ Δ 1 0
LDAA opr8a	Load Acc. A	DIR	96 dd	rPf		
LDAA opr16a		EXT	B6 hh ii	rPO		
LDAA oprx0_xysp		IDX	A6 xb	rPf		
LDAA oprx9,xysp		IDX1	A6 xb ff	rPO		
LDAA oprx16,xysp		IDX2	A6 xb ee ff	frPP		

- Above is a portion of the entry for the LDAA instruction.
- Now, we can better understand information in the HCS12 instruction sets.

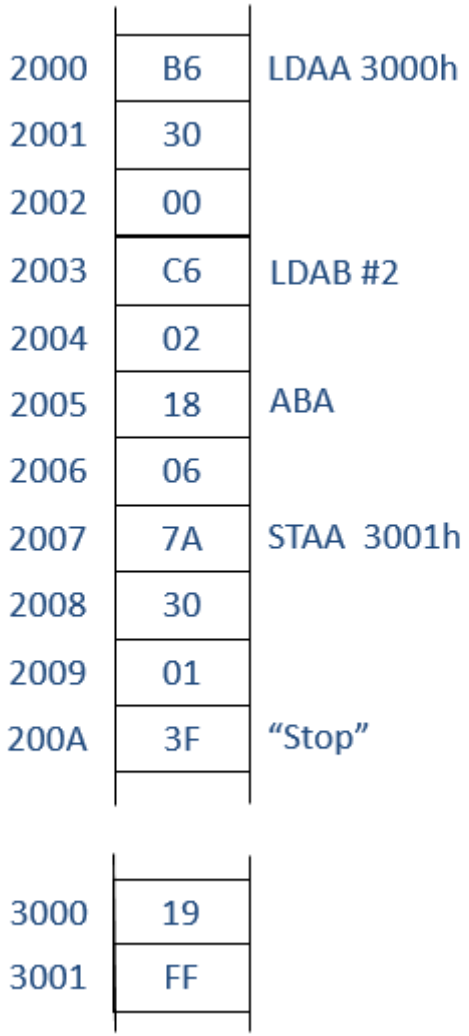
Addressing Mode Summary

How to Get an Effective Address



Program Trace

- A diagram showing the contents of the HCS12 memory which contains a program.
- A program trace shows the contents of the processor's registers as the program is executed.
- Very useful for debugging programs



Program Trace
Example

Operation: (M) ⇒ A

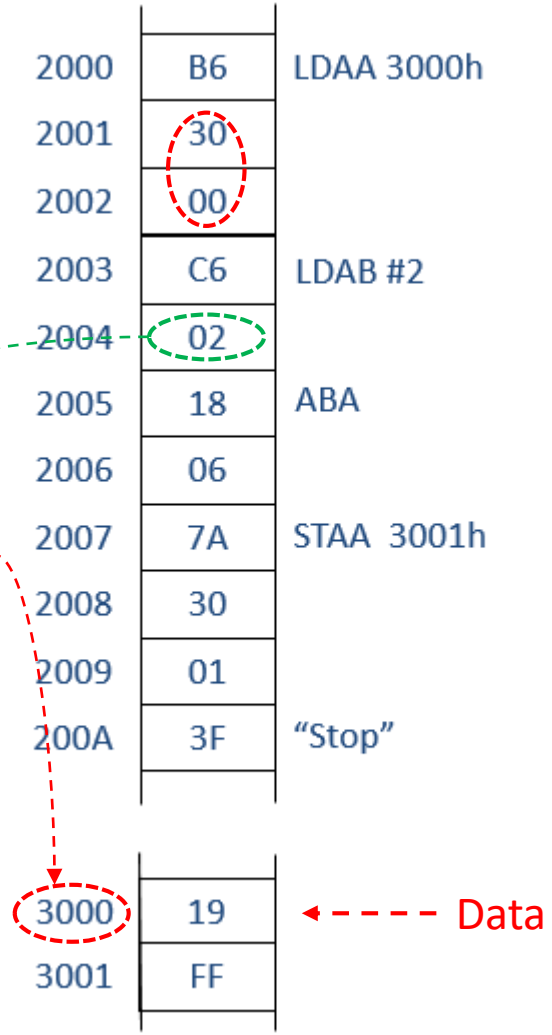
Description: Loads the content of memory location M into accumulator A. The condition codes are set according to the data.

Source Form	Address Mode	Object Code
LDAA *#opr8i*	IMM	86 ii
LDAA *opr8a*	DIR	96 dd
LDAA *opr16a*	EXT	B6 hh ll
LDAA *oprx0_xysp*	IDX	A6 xb
LDAA *oprx9,xysp*	IDX1	A6 xb ff
LDAA *oprx16,xysp*	IDX2	A6 xb ee ff
LDAA [D,*xysp*]	[D,IDX]	A6 xb
LDAA [*oprx16,xysp*]	[IDX2]	A6 xb ee ff

Source Form	Operation	Addr. Mode	Machine Coding (hex)
LDAB *#opr8i*	(M) ⇒ B	IMM	C6 ii
LDAB *opr8a*	Load Accumulator B	DIR	D6 dd
LDAB *opr16a*		EXT	F6 hh ll
LDAB *oprx0_xysp*		IDX	E6 xb
LDAB *oprx9,xysp*		IDX1	E6 xb ff
LDAB *oprx16,xysp*		IDX2	E6 xb ee ff
LDAB [D,*xysp*]		[D,IDX]	E6 xb
LDAB [*oprx16,xysp*]		[IDX2]	E6 xb ee ff

Trace Line	Address	Instruction	PC	A	B
1	2000	LDAA 3000h	2003	19	-
2	2003	LDAB #2	2005	19	02
3	2005	ABA	2007	1B	02
4	2007	STAA 3001h	200A	1B	02
5	200A	“stop”	-	-	-

Stay the same



Questions?

Wrap-up
What we've learned

- Five addressing modes
- Program trace

What to Come

- Unconditional branches
- Relative addressing mode