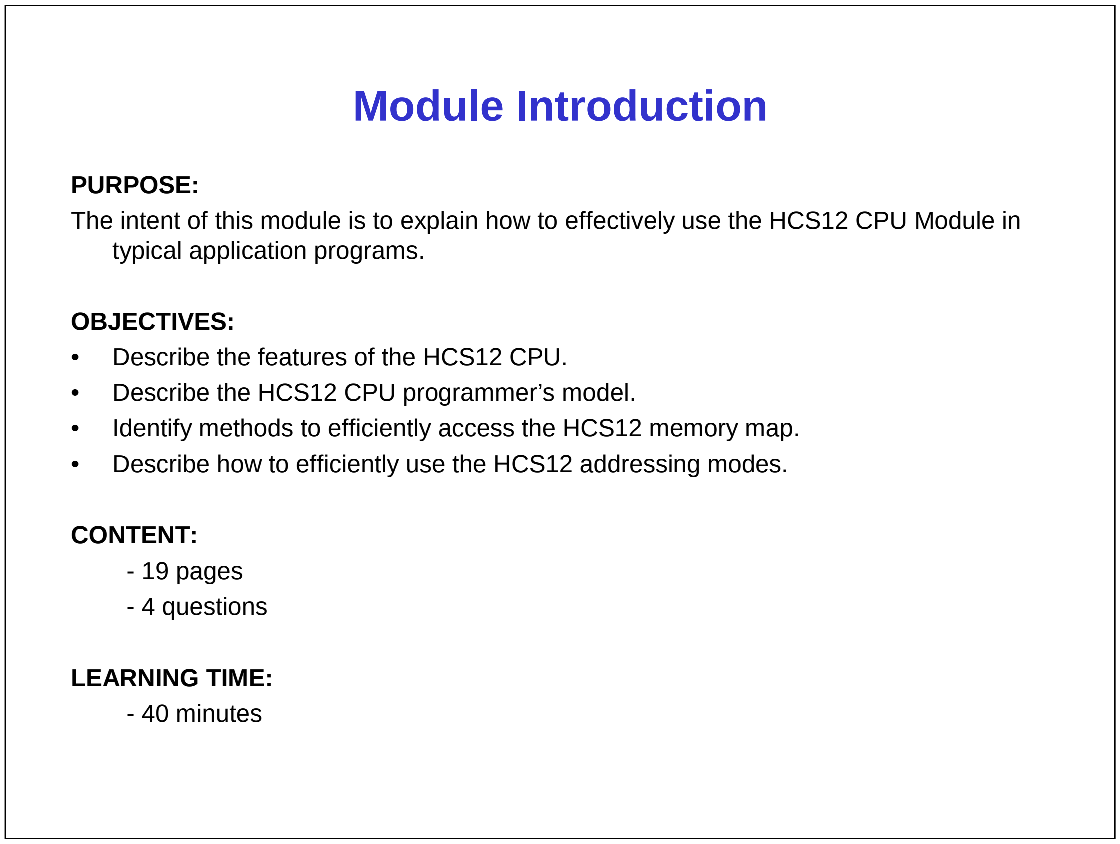


Module Introduction

PURPOSE:

The intent of this module is to explain how to effectively use the HCS12 CPU Module in typical application programs.

OBJECTIVES:

- Describe the features of the HCS12 CPU.
- Describe the HCS12 CPU programmer's model.
- Identify methods to efficiently access the HCS12 memory map.
- Describe how to efficiently use the HCS12 addressing modes.

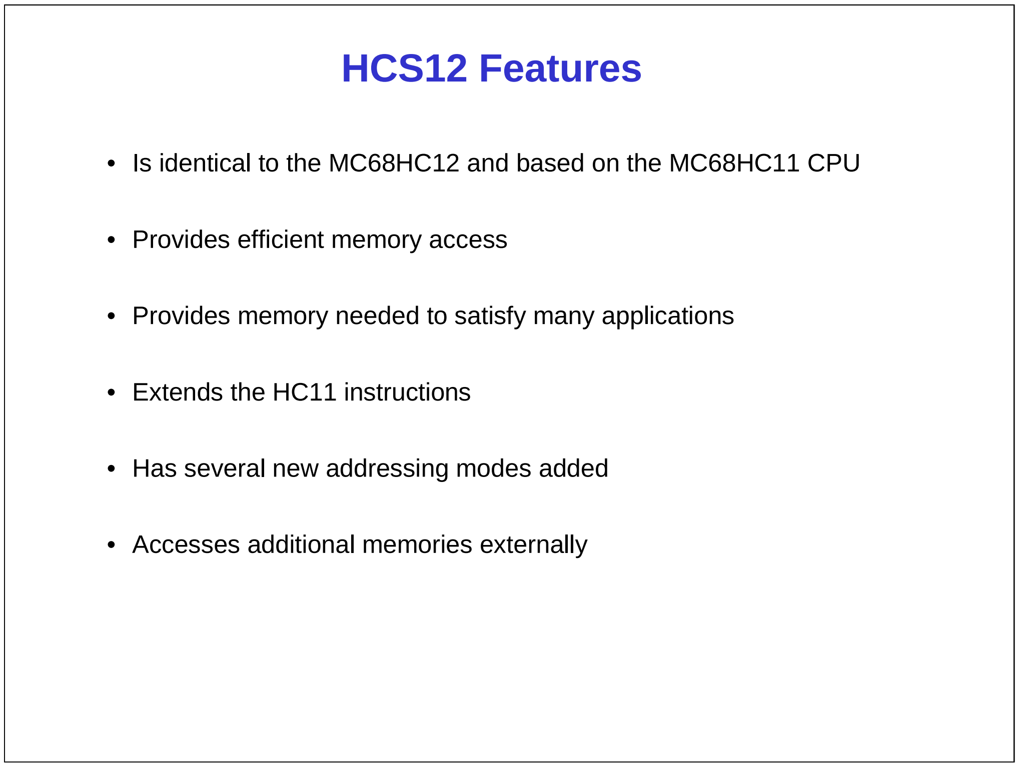
CONTENT:

- 19 pages
- 4 questions

LEARNING TIME:

- 40 minutes

The intent of this module is to explain how to effectively use the HCS12 CPU Module in typical application programs. You will become familiar with the HCS12 CPU features and programmer's model, and methods to efficiently access the HCS12 memory map. You will also explore examples of the new indexed addressing modes available in the HCS12 instruction set.



HCS12 Features

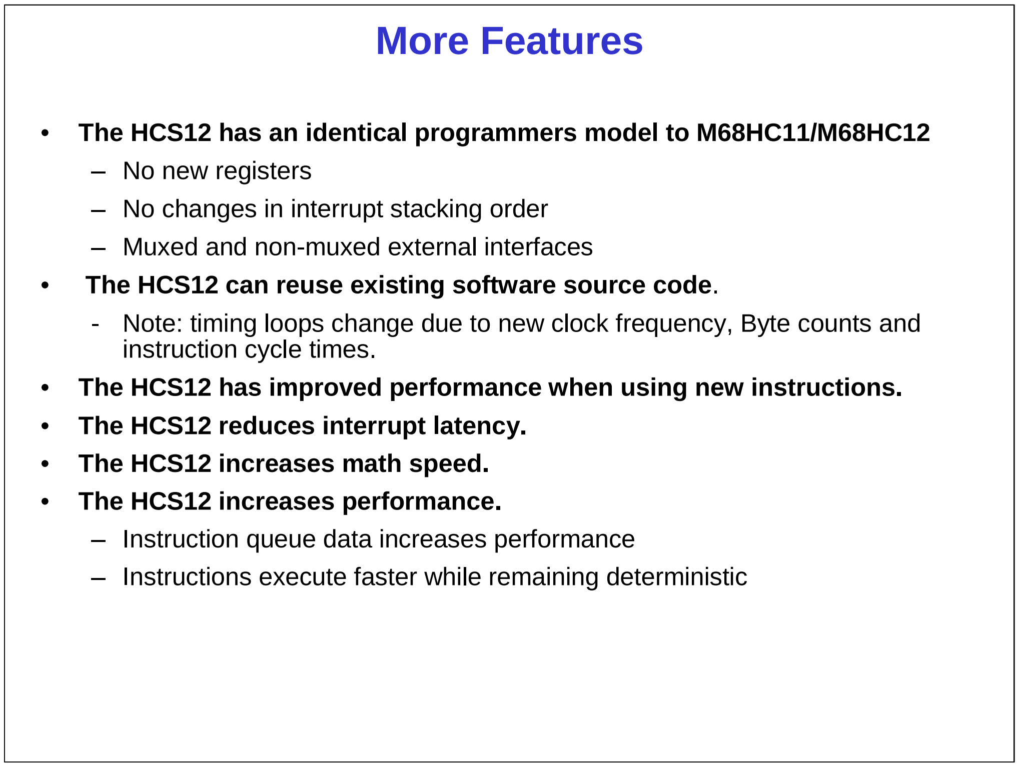
- Is identical to the MC68HC12 and based on the MC68HC11 CPU
- Provides efficient memory access
- Provides memory needed to satisfy many applications
- Extends the HC11 instructions
- Has several new addressing modes added
- Accesses additional memories externally

Here is an overview of the HCS12 CPU architecture. The HCS12 CPU is identical to the MC68HC12 and is referred to in most documents as CPU12. Another factor to consider is that CPU12 is upward compatible with the MC68HC11 CPU. The MC68HC11 and the MC68HC12 will be referred to as the HC11 and HC12 respectively in this module.

Like the HC12, the HCS12 CPU maps all registers and peripherals into a single linear address space, providing efficient memory access. Since the HCS12 family supports up to 1 mega byte of address space, a paging scheme has been implemented to access as many as 64-16K page windows. This provides system designers with the memory needed to satisfy many applications.

The HCS12 CPU instruction set extends the HC11 instructions for data movement, data manipulation, enhanced arithmetical operation, and branching and control logic. In addition, the HCS12 model adds several new addressing modes for a total of sixteen addressing modes. Although most of the HCS12 family members have on-chip Flash and RAM, which are ideal for single chip solutions for many applications, the system designer can configure the device to access additional memories externally.

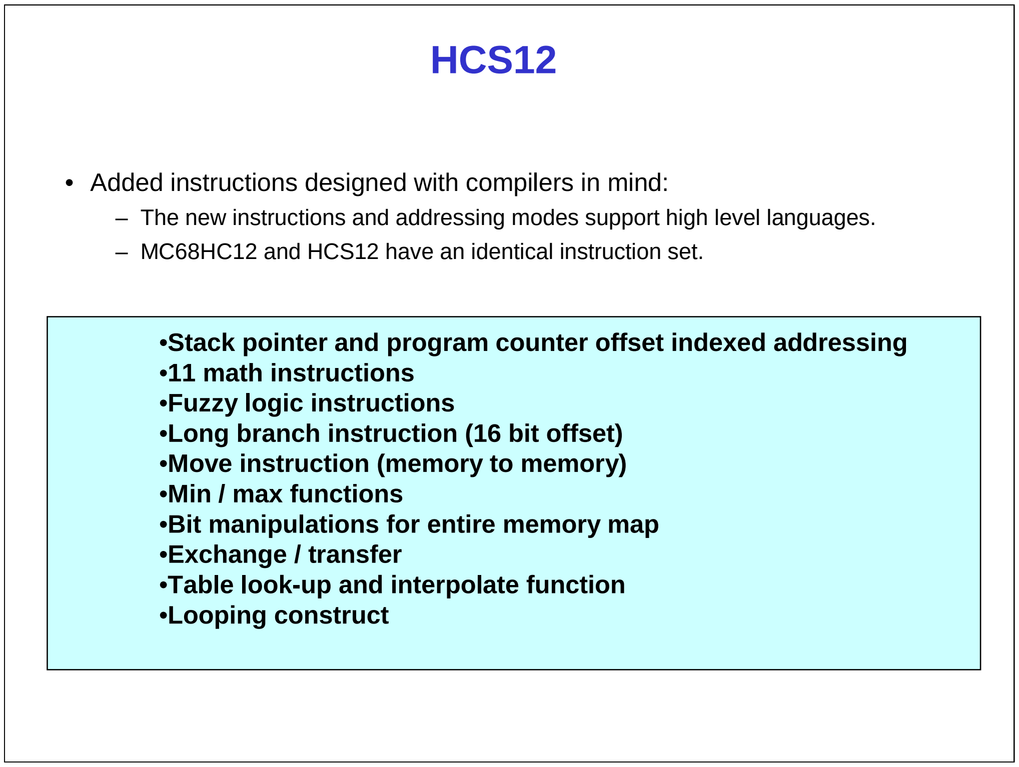
Click “More Features” to learn more about the HCS12 CPU.



More Features

- **The HCS12 has an identical programmers model to M68HC11/M68HC12**
  - No new registers
  - No changes in interrupt stacking order
  - Muxed and non-muxed external interfaces
- **The HCS12 can reuse existing software source code**.
  - Note: timing loops change due to new clock frequency, Byte counts and instruction cycle times.
- **The HCS12 has improved performance when using new instructions.**
- **The HCS12 reduces interrupt latency.**
- **The HCS12 increases math speed.**
- **The HCS12 increases performance.**
  - Instruction queue data increases performance
  - Instructions execute faster while remaining deterministic

[This is a reference page the “More Features”]



HCS12

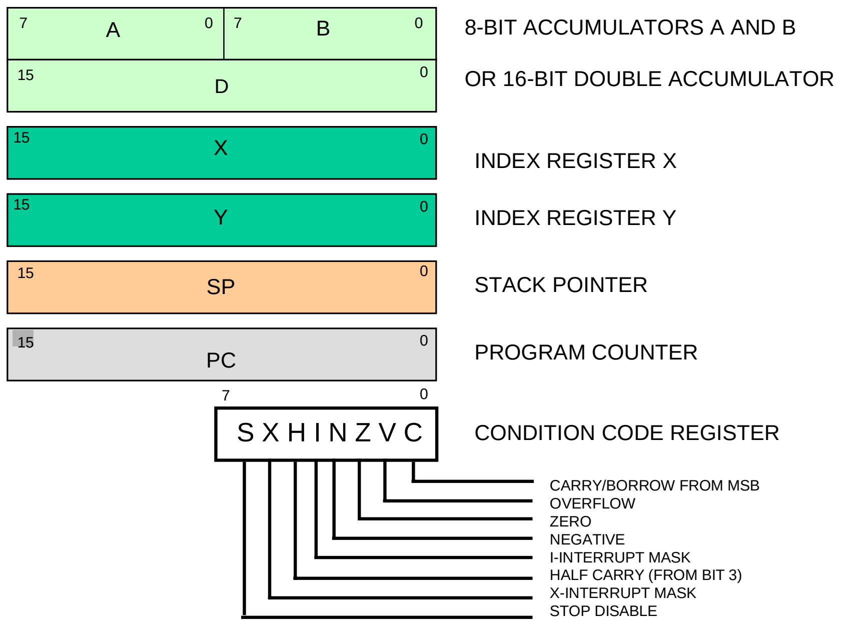
- Added instructions designed with compilers in mind:
  - The new instructions and addressing modes support high level languages.
  - MC68HC12 and HCS12 have an identical instruction set.

•Stack pointer and program counter offset indexed addressing
•11 math instructions
•Fuzzy logic instructions
•Long branch instruction (16 bit offset)
•Move instruction (memory to memory)
•Min / max functions
•Bit manipulations for entire memory map
•Exchange / transfer
•Table look-up and interpolate function
•Looping construct

The CPU12 is a high-speed, 16-bit processing unit. It has full 16-bit data paths and internal registers for high-speed extended math instructions. The instruction set is a proper superset of the M68HC11 instruction set. The CPU12 allows instructions with odd byte counts, including many single-byte instructions. This provides efficient use of ROM space. An instruction pipe buffers program information so the CPU always has immediate access to at least three bytes of machine code at the start of every instruction. The CPU12 also offers an extensive set of indexed addressing capabilities. The HCS12 includes new instructions and addressing modes to support high level languages.

There are a number of features included in the HCS12 CPU for efficiently implementing code that is written in a high-level language. CPU12 has a full set of 8 and 16-bit mathematical instructions such as 16x16 signed and unsigned multiplies and divides. CPU12 also includes a set of efficient fuzzy logic instructions, and table lookup and interpolate. It also includes many instruction that support looping construct.

Programming Model



Next, let’s take a closer look at the HCS12 CPU architecture with a review of the programmer’s model.

This is a programmer's view of the CPU registers. Here, you can see the A, B, and D accumulators. They are used to hold the operands and results of arithmetic calculations or data manipulations. A and B are two 8-bit accumulators, while D is a rich set of 16-bit operations that uses the concatenation of A:B as a 16-bit accumulator.

Index registers X and Y are used for the indexed addressing mode. They are useful for data movement and in cases where operands from two separate tables are involved in calculation. For indexed mode operations, an 8-bit positive unsigned offset in the object code is added to the present contents of X or Y. This forms the address for an operand. Y-index instructions are slightly less efficient than X because of prebyte before opcode. However, it’s better than changing temps into a single index register. It is common to load one of the index registers with the beginning address of the internal register space (usually $0000). This allows indexed addressing mode to be used to access any of the internal I/O and control regs.

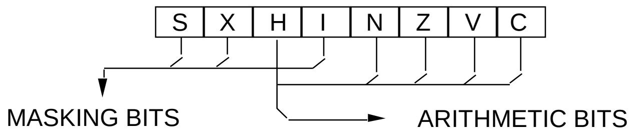
A 16-bit stack pointer (SP) implies that stack can be any size and anywhere in the 64K map. Normally, the stack pointer is initialized by one of the very first instructions in the program. Each time a byte is pushed onto the stack, the stack pointer is first decremented so that it always points to last byte pushed.

Here you can see the Program Counter (PC), a 16-bit register, that points to program memory. It holds the address of the next instruction to be executed. The PC can also reference data spaces with available HCS12 new addressing modes.

The Condition Code Register (CCR) contains 5 status indicators, 2 interrupt masking bits, and a STOP instruction disable bit.

Condition Code Register

Mouse over each letter to see its function.



- Reflect results of instruction execution.

S - Disables STOP instruction when set.

X - Masks $\overline{\text{XIRQ}}$ request when set.
- set by hardware reset, cleared by software.
- set by unmaskable $\overline{\text{XIRQ}}$

I - Masks interrupt request from all $\overline{\text{IRQ}}$ level sources (both external and internal) when set.
- set by unmasked I level request or unmasked $\overline{\text{XIRQ}}$

H - Half Carry from bit 3 to bit 4
ADD operations only

N - Negative (follows MS Bit of result)

Z - Zero result

V - 2's complement overflow indication
signed arithmetic

C - Carry/Borrow from MSB
unsigned arithmetic

Now, let’s look at the condition code register in more detail. Condition Code Flags reflect the results of arithmetic and other operations of the CPU as it performs instructions. These include H, Half Carry; N, Negative; Z, Zero; V, Overflow; and C, Carry/Borrow.

H is used only for Binary Coded Decimal (BCD) arithmetic operations and is only affected by the Add B to A (ABA), Add operation (ADD), and Add with Carry (ADC). It indicates a carry from bit 3. This flag allows the CPU to adjust the results of an 8-Bit BCD addition so it is in correct BCD format. This bit is used only by the DAA instruction to compensate the result in ACC A to correct BCD format.

N reflects the state of the most significant bit of a result. For 2’s complement, the number is negative when the MSB equals 1, and positive when MSB equals 0. It can also be used to perform some tests on this bit and make a decision depending on the result, for example BMI and BGE.

Z is when all bits of the result are zeros. The compare instructions do an internal implied subtraction, and the condition codes, including Z, reflect the results of that subtraction. INX, DEX, INY, and DEY affect the Z bit and no other bit.

V is used to indicate if a 2’s complement overflow has occurred as a result of the operation.

C is used to indicate if a carry from an addition or a borrow from a subtraction has occurred. It also acts as an error flag for multiply and divide operations. SHIFT and ROTATE instructions operate with and through the carry bit to facilitate multiple-word shift operations. Condition codes are automatically updated by almost all instructions except Pushes, Pulls, ABX, ABY and 16-bit transfers and exchanges.

STOP disable is used to allow or disallow the STOP instruction. The STOP instruction causes the oscillator to stop, therefore the user can set the S bit in the CCR to disallow the STOP instruction. If for some reason the STOP instruction is encountered while the S bit is set, then it will treat it as a NOP and normal flow of the program continues.

I-bit is the mask bit for all maskable interrupt request sources. While I bit is set, interrupts can become pending and are remembered. However, the CPU operation continues uninterrupted until the I bit is cleared. After the Hardware Reset I bit is set, and it can be cleared by software. When a maskable interrupt occurs, the condition code register (CCR) is saved on the stack and then the I bit is set to mask other interrupts that may occur during the servicing of this interrupt. The CCR is restored to it’s original value upon returning from the interrupt service routine. Restoring the CCR will re-enable interrupts.

X bit is the mask bit for non-maskable interrupts, (XIRQ). This bit masks or disables interrupts associated with the XIRQ input pin. After reset, this bit is automatically set to prevent non-maskable interrupts from being recognized by the CPU until it is cleared expicitly by software. Once it’s cleared, this bit can not be set again until another reset condition occurs.
Mouse over each letter in the bar to learn more about condition code registers.

Question

Which of the statements below represent features of the HCS12? Select all options that apply.

It provides efficient memory access.

It is identical to the MC68HC12 and based on the MC68HC11 CPU.

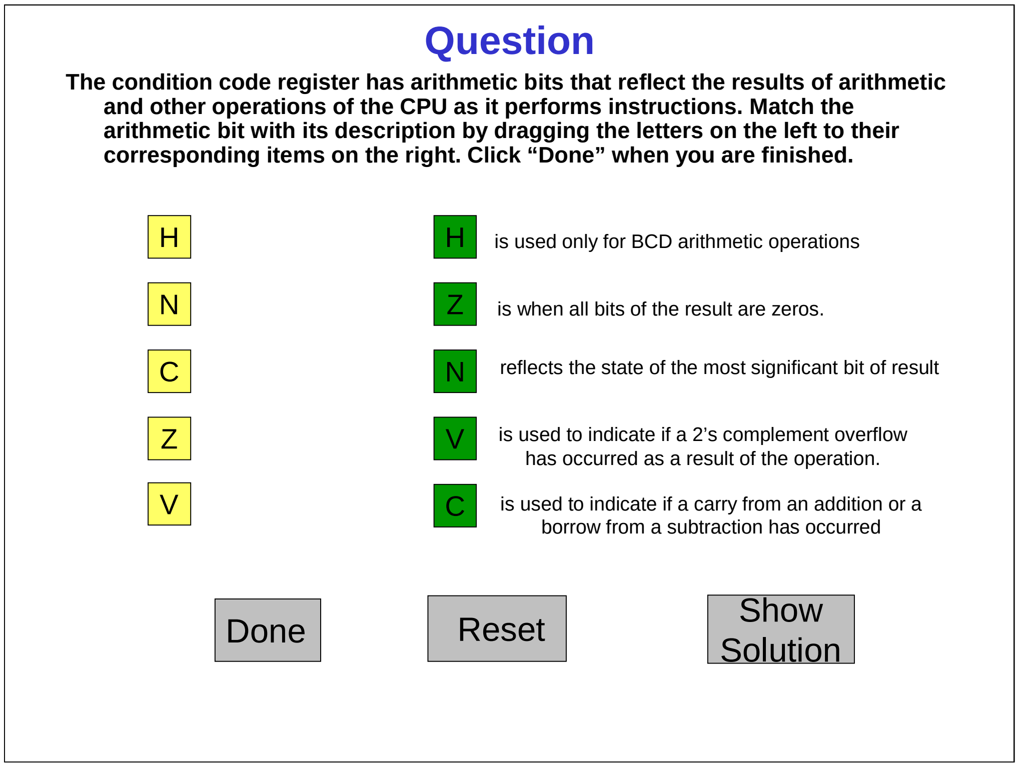
It extends the HC11 instructions.

It can reuse existing software source code.

It has reduced interrupt latency.

Consider this question regarding various features of the HCS12.

The HCS12 maps all registers and peripherals into a single linear address space, providing efficient memory access. It is identical to the MC68HC12 and is based on the MC68HC11 CPU. The HCS12 instruction set extends the HC11 instructions for data movement, data manipulation, enhanced arithmetical operation and branching and control logic. The HCS12 can also reuse existing software source code and has reduced interrupt latency.



Question

The condition code register has arithmetic bits that reflect the results of arithmetic and other operations of the CPU as it performs instructions. Match the arithmetic bit with its description by dragging the letters on the left to their corresponding items on the right. Click “Done” when you are finished.

H	H	is used only for BCD arithmetic operations
N	Z	is when all bits of the result are zeros.
C	N	reflects the state of the most significant bit of result
Z	V	is used to indicate if a 2’s complement overflow has occurred as a result of the operation.
V	C	is used to indicate if a carry from an addition or a borrow from a subtraction has occurred

Let’s review the condition code register.

These condition code registers include H, Half Carry; N, Negative; C, Carry/Borrow, V, Overflow; and Z, Zero.

HCS12 Addressing Modes

HC11 Addressing Modes

INHERENT	CLRB	
IMMEDIATE	LDAA	$$12
EXTENDED	LDAA	$4000
DIRECT	LDAA	$50
RELATIVE	BNE	LOOP
INDEXED	LDAB	$10,X

Now, let’s look at addressing modes. Addressing modes determine how the CPU accesses memory locations to be operated upon. The HCS12 CPU includes all of the addressing modes of the M68HC11 CPU as well as several new forms of indexed addressing.

Listed are the original HC11 addressing modes with example instructions. These modes include inherent, immediate, extended, direct, relative,and indexed addressing modes. These addressing modes can be used efficiently in both the HC12 and the HCS12 as well.

Inherent instructions have no operand fetch as the operand is defined in the 8-bit opcode. Clear Accumulator (CLRB) is a single cycle instruction for CPU12. CLRB takes one E-clock cycle to execute.

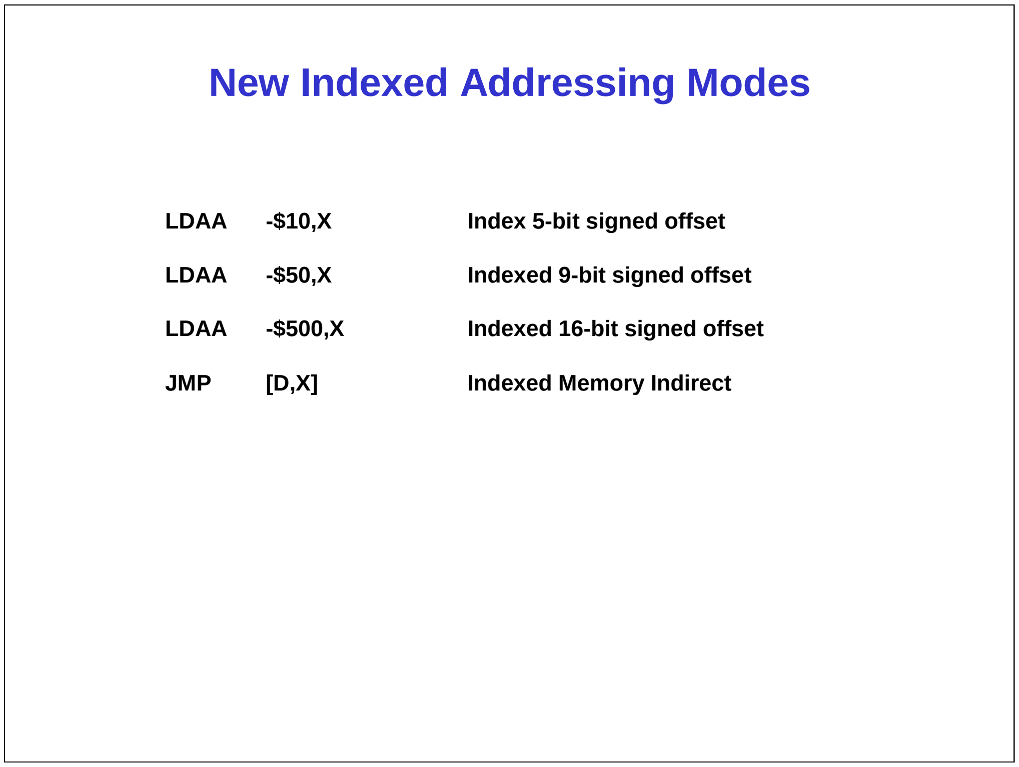
Immediate instructions have the 8-bit or 16-bit operand immediately following the opcode. Load Accumulator A (LDAA) with $12 is a two-byte instruction that executes in 2 bus cycles.

Extended addressing instructions provide absolute addressing to any location in the 64K memory map without paging. They require three bytes total for the opcode plus the 16-bit address of the operand.

Direct addressing instructions have the 8-bit address of the operand immediately following the opcode. Therefore, direct instructions access the first 256 bytes of memory. Direct addressing refers to this type of access as direct page or page zero addressing. Load accumulator with the operand at memory location $50 is a two-byte instruction that executes in 3 bus cycles.
Most assemblers will automatically use the shorter direct mode for any access to the first 256 bytes of the memory map.

All conditional branch instructions use relative addressing. If the branch condition is true, the PC is added to the signed byte immediately following the branch opcode. This gives a -128 to +127 byte range for branching.

Next, let’s take a look at indexed addressing.



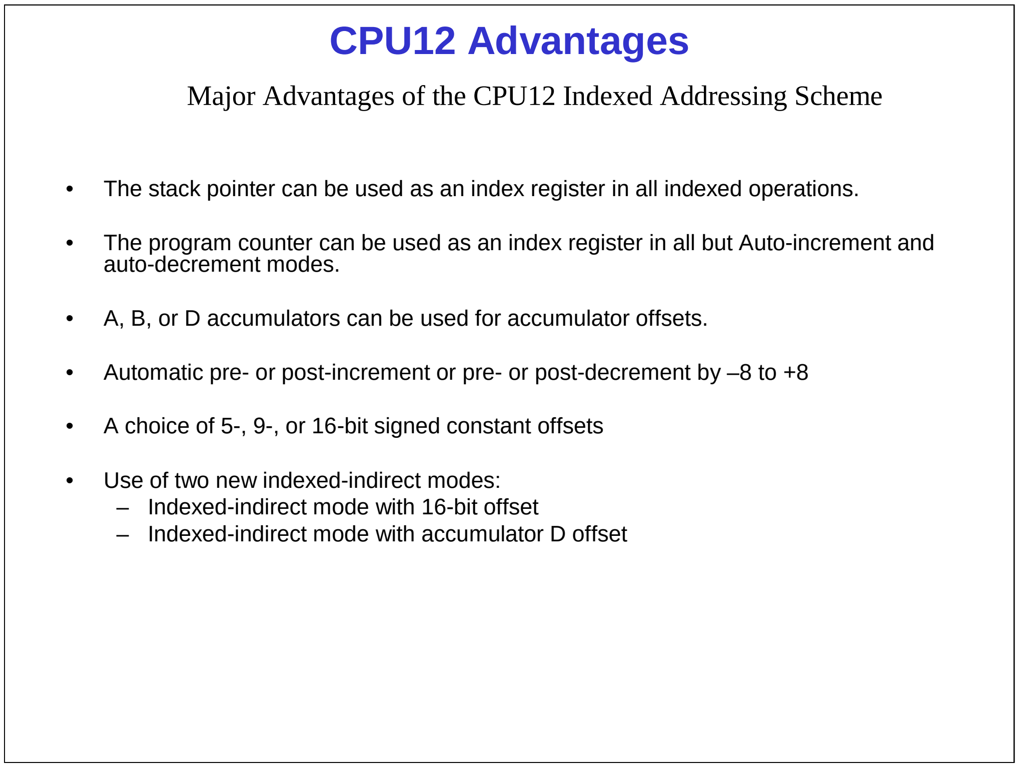
New Indexed Addressing Modes

LDAA	-$10,X	Index 5-bit signed offset
LDAA	-$50,X	Indexed 9-bit signed offset
LDAA	-$500,X	Indexed 16-bit signed offset
JMP	[D,X]	Indexed Memory Indirect

The HCS12 has a variety of indexed addressing modes. These addressing modes provide the CPU with a sometimes dramatic improvement in code efficiency. This is compared to architectures with just indirect addressing or indirect address with another offset register.

The indexed addressing modes are key to efficiently addressing tables and other data structures. Indexed addressing with offset is what most other architectures refer to as indirect addressing. The value in the index register is the address, or pointer, of the operand. The offset portion can be used to select a particular element in a table or data array.

To learn some major advantages of the CPU12 indexed addressing scheme, click “CPU12 Advantages”.



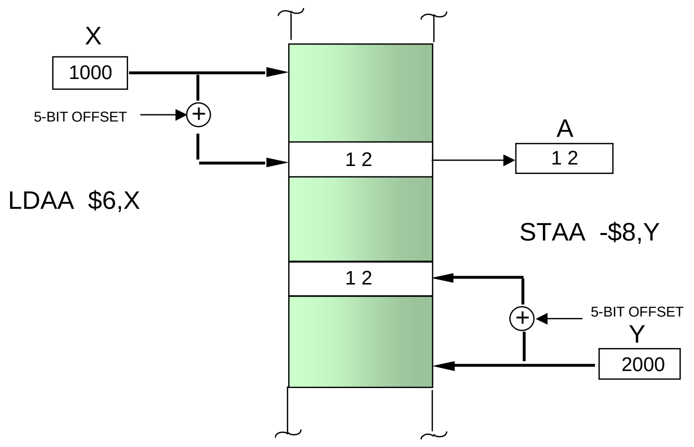
CPU12 Advantages

Major Advantages of the CPU12 Indexed Addressing Scheme

- The stack pointer can be used as an index register in all indexed operations.
- The program counter can be used as an index register in all but Auto-increment and auto-decrement modes.
- A, B, or D accumulators can be used for accumulator offsets.
- Automatic pre- or post-increment or pre- or post-decrement by –8 to +8
- A choice of 5-, 9-, or 16-bit signed constant offsets
- Use of two new indexed-indirect modes:
  - Indexed-indirect mode with 16-bit offset
  - Indexed-indirect mode with accumulator D offset

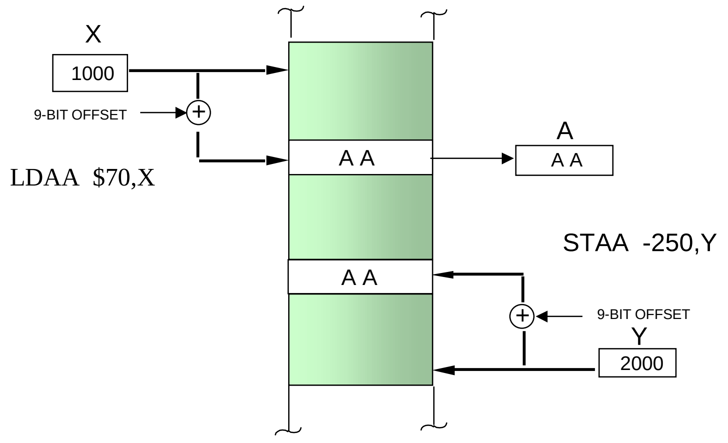
[This page is a reference to the “CPU12 Advantages”]

Indexed 5-Bit Signed Offset



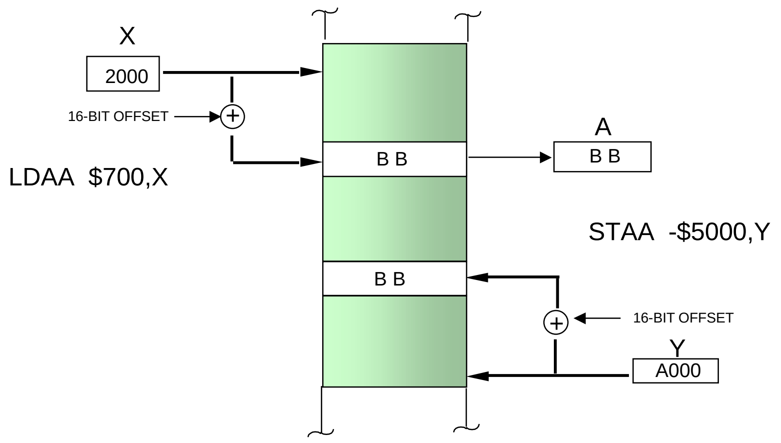
Now, let’s look at some examples of the new indexed addressing modes available in the HCS12 instruction set. The first indexed addressing mode shown here uses an efficient short 5-bit signed offset allowing the CPU to access the memory location pointed to by the index register +15 and –16 byte location. The short 5-bit offset is added to the base index register (X, Y, SP, or PC) to form the effective address of memory location that will be affected by the instruction.

Indexed – Accumulator Offset



The second indexed addressing mode shown here uses 9-bit signed offset which is added to the index register (X, Y, SP or PC) allowing the CPU to access the memory location pointed to by the index register +255 and -256 byte location.

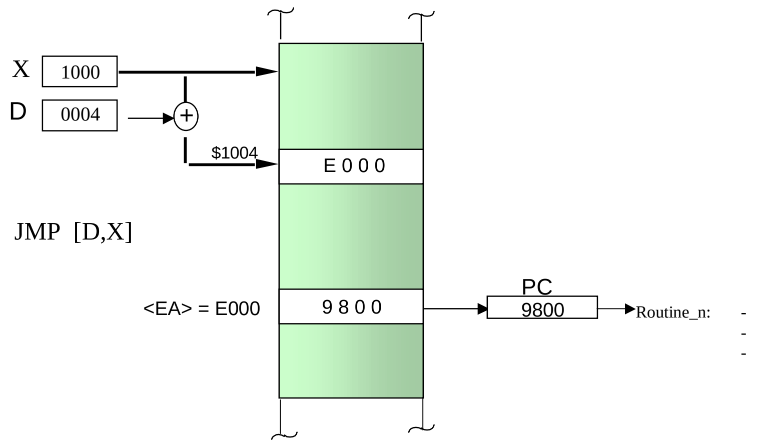
Indexed 16-Bit Offset (IDX2)



The third indexed addressing mode shown here uses 16-bit signed offset which is added to the index register (X, Y, SP or PC) allowing the CPU to access the memory location pointed to by the index register with an offset of up to +32K and –32K byte locations. This addressing mode, with 16-bit offset, allows the CPU to index to any memory location in the 64K memory map.

Indexed - D - Indirect ([D,IDX])

Mouse over the diagram to learn more.



- D- Accumulator added to base register to form <ea> of indirect address.
- X, Y, SP or PC register can be used for indexing.
- Offset range + or - 32KBytes from base register.

The HCS12 is a very powerful 16 bit microcontroller with a limited number of pointer registers. The limitation in the number of registers is inherited from the HC11 in order to maintain software compatibility. However, the indexed indirect addressing modes gives the HCS12 an unlimited number of memory pointers. This addressing mode in other architectures is sometimes referred to as memory indirect addressing mode.

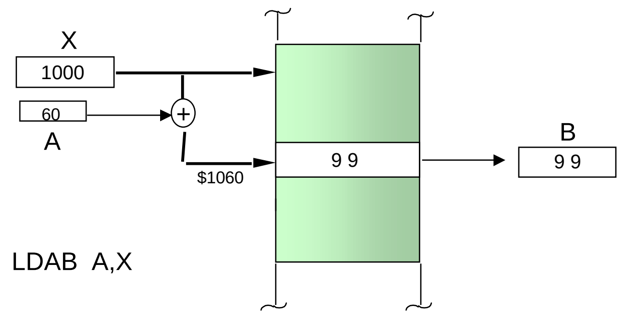
This addressing mode calculates the effective address in the X and D registers to select a pointer from a table of multiple pointers. Sometimes an I/O device may require multiple services inside an interrupt service routine. For example, the SCI can generate up to 4 different interrupts that use the same vector in the MCU vector table. The software has to determine which one of the four routines needs to be executed to handle a particular SCI interrupt event. The user may poll the SCI status register to find the cause of the interrupt. The user then vectors to a particular routine within the interrupt handler.

To support this concept, the user builds a jump table allowing access to a particular pointer to be selected. This will point to a unique routine depending on the current pending event being requested from the SCI. In this example, the index register X points to the jump table base address and the D accumulator is used as an offset to point to a desired pointer. Mouse over the diagram to learn more.

Indexed – Accumulator Offset

Accumulator Offset Indexed Addressing

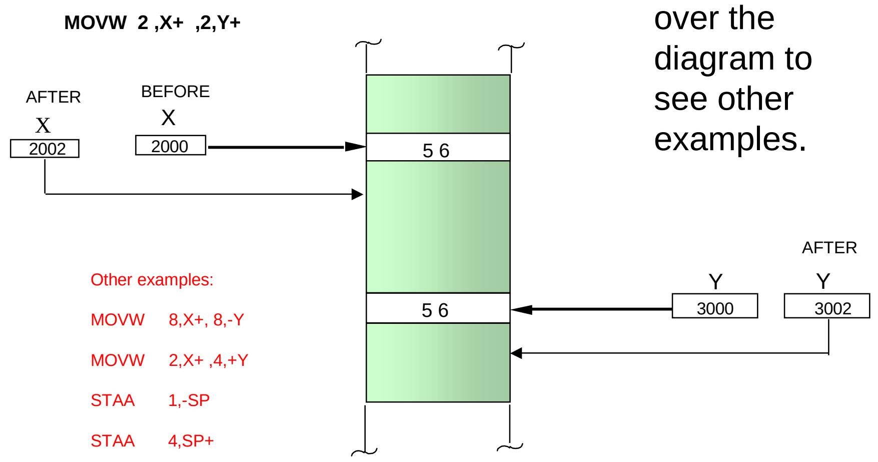
Indexed- Accumulator Offset



In this indexed addressing mode, the effective address is the sum of the values in the base index register and an unsigned offset in one of the accumulators. The value in the index register itself is not changed. The index register can be X, Y, SP, or PC. The accumulator can be either of the 8-bit accumulators (A or B) or the 16-bit D accumulator. The offset range can be up to + or – 32 Kbytes from base register.

In this example, the LDAB A,X instruction loads accumulator B from the memory location pointed by index register X plus 8-bit offset in accumulator A.

Pre/Post Decrement/Increment



Other examples:

MOVW	8,X+, 8,-Y
MOVW	2,X+ ,4,+Y
STAA	1,-SP
STAA	4,SP+

Mouse over the diagram to see other examples.

This indexed addressing mode provides four ways to automatically change the value in a base index register as a part of instruction execution. The index register can be incremented or decremented by an integer value either before or after indexing takes place. The base index register may be X, Y, or SP. Note that auto-modify modes would not make sense on PC.

Pre-decrement and pre-increment versions of the addressing mode adjust the value of the index register before accessing the memory location affected by the instruction. The index register retains the changed value after the instruction executes. Post-decrement and post-increment versions of the addressing mode use the initial value in the index register to access the memory location affected by the instruction. Then, they change the value of the index register.

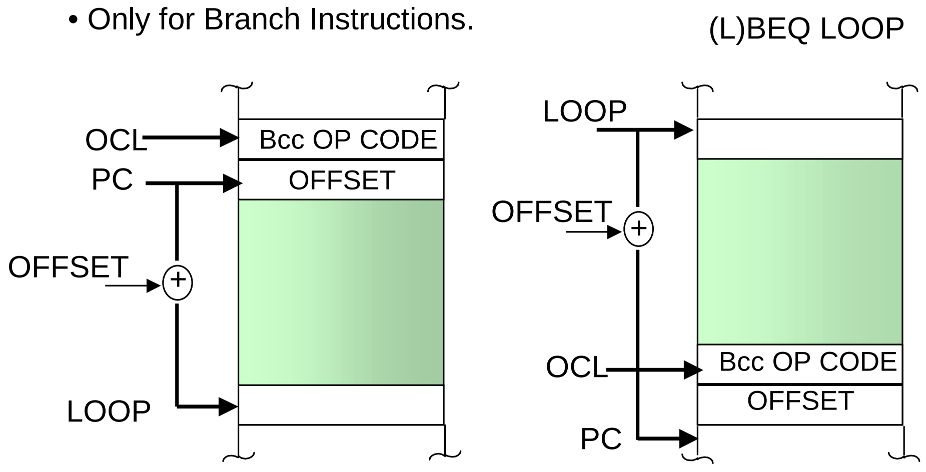
The CPU12 allows the index register to be incremented or decremented by any integer value in the ranges –8 through –1 or 1 through 8. The value need not be related to the size of the operand for the current instruction. These instructions can be used to incorporate an index adjustment into an existing instruction, rather than using an additional instruction and increasing execution time. This addressing mode is also used to perform operations on a series of data structures in memory.

Mouse over the diagram to see other examples.

Relative Addressing

Mouse over either diagram to learn more.

- Only for Branch Instructions.



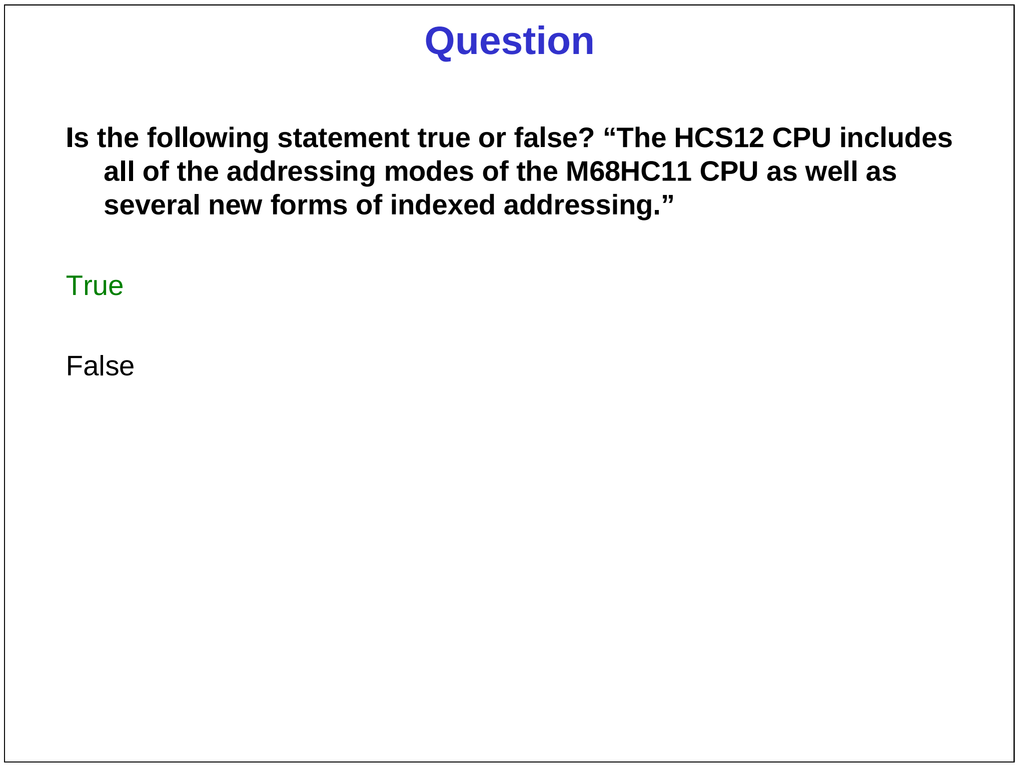
- Branch Instructions are 2 or 4 bytes in length.
- All Branches are taken from the next instruction address

  (Destination of branch is calculated by adding signed offset byte to OCL +2 OR +4)

The Program Counter (PC) relative addressing mode is used only by branch instructions. Short and long conditional branch instructions use relative addressing mode exclusively. However, branching versions of bit manipulation instructions (branch if bits set (BRSET) and branch if bits cleared (BRCLR) use multiple addressing modes, including relative mode.

Short branch instructions consist of an 8-bit opcode and a signed 8-bit offset that is contained in the byte that follows the opcode. Long branch instructions allows branching with a 16-bit offset contained in the two bytes that follow the opcode.

Each conditional branch instruction tests certain bits in the condition code register to determine if the specified condition is met. If the bits are in a specified state, the offset is added to the address of the next memory location to point to the branch target address, meaning condition is met. If the bits are not in the specified state, execution continues with the instruction immediately following the branch instruction, meaning condition tested was not met. Roll your mouse pointer over either diagram to learn more.



Question

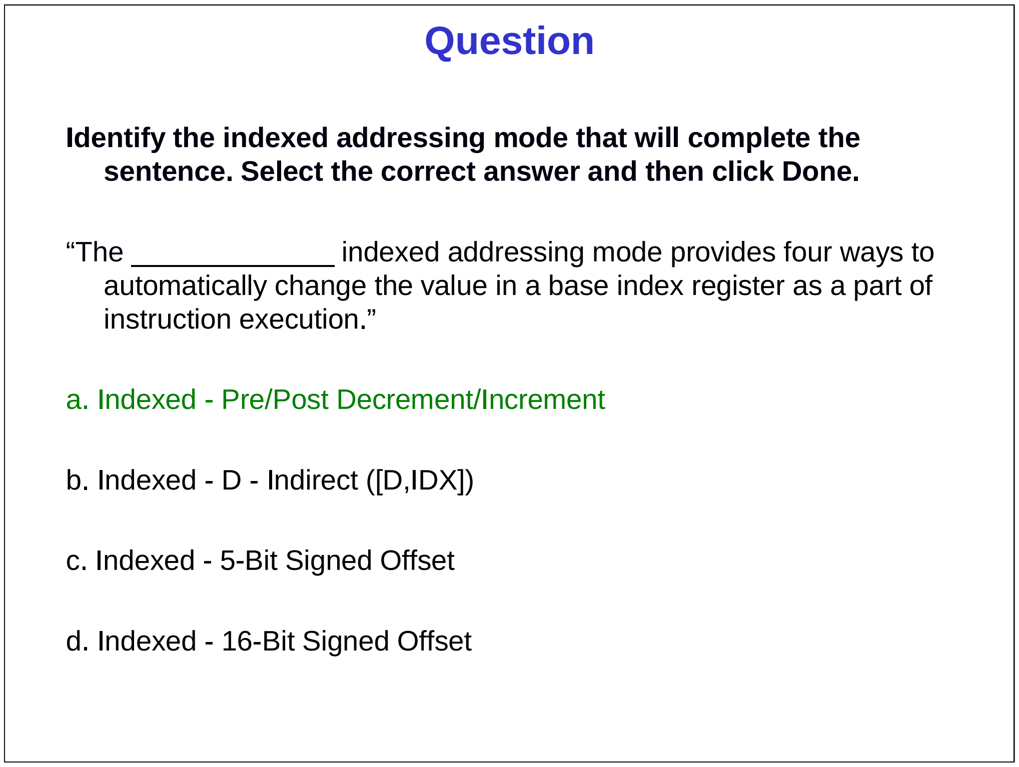
Is the following statement true or false? “The HCS12 CPU includes all of the addressing modes of the M68HC11 CPU as well as several new forms of indexed addressing.”

True

False

Consider this question regarding addressing modes.

Addressing modes determine how the CPU accesses memory locations to be operated upon. The HCS12 CPU includes all of the addressing modes of the M68HC11 CPU as well as several new forms of indexed addressing.



Question

Identify the indexed addressing mode that will complete the sentence. Select the correct answer and then click Done.

“The _____________ indexed addressing mode provides four ways to automatically change the value in a base index register as a part of instruction execution.”

a. Indexed - Pre/Post Decrement/Increment

b. Indexed - D - Indirect ([D,IDX])

c. Indexed - 5-Bit Signed Offset

d. Indexed - 16-Bit Signed Offset

Let’s see if you can remember the different indexed addressing modes.

The Pre/Post Decrement/Increment indexed addressing mode provides four ways to automatically change the value in a base index register as a part of instruction execution. The index register can be incremented or decremented by an integer value either before or after indexing takes place.

Module Summary

- HCS12 Features
- Programming Model
- Condition Code Register
- HCS12 Addressing Modes

Now that you have completed this module, you should be able to describe the features of the HCS12 CPU. You should be able to describe the programming model, and know more detailed information about the condition code register. You should also be able to identify methods to efficiently access the HCS12 memory map. Finally, you should be able to describe how to efficiently use the HCS12 addressing modes.