The assembly line of code within the set of quotation marks has the same format as a valid assembly statement. Note that if the instruction has a label, the first character of the label must start after the first quotation mark so that it is in column 1. The assembly statement should be valid since the compiler does not check it for syntax error but copies it directly into the compiled output file. If the assembly statement has a syntax error, the assembler would detect it.

Avoid using *asm* statements within a C program, especially within a linear assembly program. This is because the assembler optimizer could rearrange lines of code near the `asm` statements that may cause undesirable results.

3.12 C-CALLABLE ASSEMBLY FUNCTION

Programming examples are included later in this chapter to illustrate a C program calling an assembly function. Register B3 is preserved and is used to contain the return address of the calling function.

An external declaration of an assembly function called within a C program using `extern` is optional. For example,

```
extern int func();
```

is optional with the assembly function `func` returning an integer value.

3.13 TIMERS

Two 32-bit timers can be used to time and count events or to interrupt the CPU. A timer can direct an external ADC to start conversion or the DMA controller to start a data transfer. A timer includes a time period register, which specifies the timer's frequency; a timer counter register, which contains the value of the incrementing counter; and a timer control register, which monitors the timer's status.

3.14 INTERRUPTS

An interrupt can be issued internally or externally. An interrupt stops the current CPU process so that it can perform a required task initiated by the interrupt. The program flow is redirected to an ISR. The source of the interrupt can be an ADC, a timer, and so on. On an interrupt, the conditions of the current process must be saved so that they can be restored after the interrupt task is performed. On interrupt, registers are saved and processing continues to an ISR. Then the registers are restored.

There are 16 interrupt sources. They include two timer interrupts, four external interrupts, four McBSP interrupts, and four DMA interrupts. Twelve CPU interrupts (INT4–INT11) are available. An interrupt selector is used to choose among the 12 interrupts.

3.14.1 Interrupt Control Registers

The interrupt control registers (Appendix B) are as follows:

1. CSR (control status register): contains the global interrupt enable (GIE) bit and other control/status bits
2. IER (interrupt enable register): enables/disables individual interrupts
3. IFR (interrupt flag register): displays the status of interrupts
4. ISR (interrupt set register): sets pending interrupts
5. ICR (interrupt clear register): clears pending interrupts
6. ISTP (interrupt service table pointer): locates an ISR
7. IRP (interrupt return pointer)
8. NRP (nonmaskable interrupt return pointer)

Interrupts are prioritized, with Reset having the highest priority. The reset interrupt and nonmaskable interrupt (NMI) are external pins that have the first and second highest priority, respectively. The interrupt enable register (IER) is used to set a specific interrupt and can check if and which interrupt has occurred from the interrupt flag register (IFR).

NMI is nonmaskable, along with Reset. NMI can be masked (disabled) by clearing the nonmaskable interrupt enable (NMIE) bit within CSR. It is set to zero only upon reset or upon a nonmaskable interrupt. If NMIE is set to zero, all interrupts INT4 through INT15 are disabled. The interrupt registers are shown in Appendix B.

The reset signal is an active-low signal used to halt the CPU, and the NMI signal alerts the CPU to a potential hardware problem. Twelve CPU interrupts with lower priorities are available, corresponding to the maskable signals INT4 through INT15. The priorities of these interrupts are: INT4, INT5, . . . , INT15, with INT4 having the highest priority and INT15 the lowest priority. For an NMI to occur, the NMIE bit must be 1 (active high). On reset (or after a previously set NMI), the NMIE bit is cleared to zero so that a reset interrupt may occur.

To process a maskable interrupt, the GIE bit within the control status register (CSR) and the NMIE bit within the IER are set to 1. GIE is set to 1 with bit 0 of CSR set to 1, and NMIE is set to 1 with bit 1 of IER set to 1. Note that CSR can be ANDed with –2 (using 2's complement, the LSB is 0, while all other bits are 1's) to set the GIE bit to 0 and disable maskable interrupts globally.

The interrupt enable (IE) bit corresponding to the desirable maskable interrupt is also set to 1. When the interrupt occurs, the corresponding IFR bit is set to 1 to show the interrupt status. To process a maskable interrupt, the following apply:

1. The GIE bit is set to 1.
2. The NMIE bit is set to 1.
3. The appropriate IE bit is set to 1.
4. The corresponding IFR bit is set to 1.

TABLE 3.5 Interrupt Service Table

Interrupt	Offset
RESET	000h
NMI	020h
Reserved	040h
Reserved	060h
INT4	080h
INT5	0A0h
INT6	0C0h
INT7	0E0h
INT8	100h
INT9	120h
INT10	140h
INT11	160h
INT12	180h
INT13	1A0h
INT14	1C0h
INT15	1E0h

For an interrupt to occur, the CPU must not be executing a delay slot associated with a branch instruction.

The interrupt service table (IST) shown in Table 3.5 is used when an interrupt begins. Within each location is an FP associated with each interrupt. The table contains 16 FPs, each with eight instructions. The addresses on the right side correspond to an offset associated with each specific interrupt. For example, the FP for interrupt INT11 is at a base address plus an offset of 160 h. Since each FP contains eight 32-bit instructions (256 bits) or 32 bytes, each offset address in the table is incremented by 20 h = 32.

The reset FP must be at address 0. However, the FPs associated with the other interrupts can be relocated. The relocatable address can be specified by writing this address to the interrupt service table base (ISTB) register of the interrupt service table pointer (ISTP) register, shown in Figure B.7. On reset, ISTB is zero. For relocating the vector table, the ISTP is used; the relocatable address is ISTB plus the offset.

3.14.2 Interrupt Acknowledgment

The signals IACK and INUMx (INUM0 through INUM3) are pins on the C6x that acknowledge that an interrupt has occurred and is being processed. The four INUMx signals indicate the number of the interrupt being processed. For example,

```
INUM3 = 1 (MSB), INUM2 = 0, INUM1 = 1, INUM0 = 1 (LSB)
```

correspond to $(1011)_b = 11$, indicating that INT11 is being processed.

The IE11 bit is set to 1 to enable INT11. The IFR can be read to verify that bit IF11 is set to 1 (INT11 enabled). Writing a 1 to a bit in the interrupt set register (ISR) causes the corresponding interrupt flag to be set in IFR, whereas a 0 to a bit in the interrupt clear register (ICR) causes the corresponding interrupt to be cleared.

All interrupts remain pending while the CPU has a pending branch instruction. Since a branch instruction has five delay slots, a loop smaller than six cycles is non-interruptible. Any pending interrupt will be processed as long as there are no pending branches to be completed. Additional information can be found in Ref. 6.

3.15 MULTICHANNEL BUFFERED SERIAL PORTS

Two McBSPs are available. They provide an interface to inexpensive (industry standard) external peripherals. McBSPs have features such as full-duplex communication, independent clocking and framing for receiving and transmitting, and direct interface to AC97 and IIS compliant devices. They allow several data sizes between 8 and 32 bits. Clocking and framing associated with the McBSPs for input and output are discussed in Ref. 7.

External data communication can occur while data are being moved internally. Figure 3.4 shows an internal block diagram of a McBSP. The data transmit (DX) and data receive (DR) pins are used for data communication. Control information (clocking and frame synchronization) is through CLKX, CLKR, FSX, and FSR. The CPU or DMA controller reads data from the data receive register (DRR) and writes data to be transmitted to the data transmit register (DXR). The transmit shift register (XSR) shifts these data to DX. The receive shift register (RSR) copies the data received on DR to the receive buffer register (RBR). The data in RBR are then copied to DRR to be read by the CPU or the DMA controller.

Other registers—the serial port control register (SPCR), receive/transmit control register (RCR/XCR), receive/transmit channel enable register (RCER/XCER), pin control register (PCR), and sample rate generator register (SRGR)—support further data communication [7].

The two McBSPs are used for input and output through the onboard codec. McBSP0 is used for control and McBSP1 for transmitting and receiving data.

3.16 DIRECT MEMORY ACCESS

Direct memory access (DMA) allows for the transfer of data to and from internal memory or external devices without intervention from the CPU [7]. Sixteen enhanced DMA channels (EDMA) can be configured independently for data transfer. DMA can access on-chip memory and the EMIF, as well as the HPI. Data of different sizes can be transferred: 8-bit bytes, 16-bit half-words, and 32-bit words.

A number of DMA registers are used to configure the DMA: address (source and destination), index, count reload, DMA global data, and control registers. The source and destination addresses can be from internal program memory, internal