Lecture 13

9. Identify the addressing modes of the operands in each of the following instructions
   a. ADD #1234h
   b. ADD 1234h
   c. ADD *AR+
   d. ADD offsetreg-,*AR

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Addressing Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD #1234h</td>
<td>Immediate Addressing Mode</td>
</tr>
<tr>
<td>ADD 1234h</td>
<td>Direct Addressing Mode</td>
</tr>
<tr>
<td>ADD *AR+</td>
<td>Post Increment Indirect Addressing Mode</td>
</tr>
<tr>
<td>ADD offsetreg-,*AR</td>
<td>Pre-Sub_Offset Indirect Addressing Mode</td>
</tr>
</tbody>
</table>

2.7 Special Addressing Modes

For the implementation of some real time applications in DSP, normal addressing modes will not completely serve the purpose. Thus some special addressing modes are required for such applications.

2.7.1 Circular Addressing Mode

While processing the data samples coming continuously in a sequential manner, circular buffers are used. In a circular buffer the data samples are stored sequentially from the initial location till the buffer gets filled up. Once the buffer gets filled up, the next data samples will get stored once again from the initial location. This process can go forever as long as the data samples are processed in a rate faster than the incoming data rate.

Circular Addressing mode requires three registers viz
   a. Pointer register to hold the current location (PNTR)
   b. Start Address Register to hold the starting address of the buffer (SAR)
   c. End Address Register to hold the ending address of the buffer (EAR)
There are four special cases in this addressing mode. They are

a. \( \text{SAR} < \text{EAR} \) & updated PNTR > EAR
b. \( \text{SAR} < \text{EAR} \) & updated PNTR < SAR
c. \( \text{SAR} > \text{EAR} \) & updated PNTR > SAR
d. \( \text{SAR} > \text{EAR} \) & updated PNTR < EAR

The buffer length in the first two case will be \((\text{EAR} - \text{SAR} + 1)\) whereas for the next tow cases \((\text{SAR} - \text{EAR} + 1)\)

The pointer updating algorithm for the circular addressing mode is as shown below.

; Pointer Updating Algorithm

\[
\begin{align*}
\text{Updated PNTR} &\leftarrow \text{PNTR} \pm \text{increment} \\
\text{If} \quad \text{SAR} < \text{EAR} \\
\quad \text{And if Updated PNTR} > \text{EAR, then} & \\
\quad \quad \text{New PNTR} &\leftarrow \text{Updated PNTR} - \text{Buffer size} \\
\quad \text{And if Updated PNTR} < \text{SAR, then} & \\
\quad \quad \text{New PNTR} &\leftarrow \text{Updated PNTR} + \text{Buffer size} \\
\text{If} \quad \text{SAR} > \text{EAR} \\
\quad \text{And if Updated PNTR} > \text{SAR, then} & \\
\quad \quad \text{New PNTR} &\leftarrow \text{Updated PNTR} - \text{Buffer size} \\
\quad \text{And if Updated PNTR} < \text{EAR, then} & \\
\quad \quad \text{New PNTR} &\leftarrow \text{Updated PNTR} + \text{Buffer size} \\
\text{Else} & \\
\quad \text{New PNTR} &\leftarrow \text{Updated PNTR}.
\end{align*}
\]
Four cases explained earlier are as shown in the figure 2.12.

Fig 2.12 Special Cases in Circular Addressing Mode
11. A DSP has a circular buffer with the start and the end addresses as 0200h and 020Fh respectively. What would be the new values of the address pointer of the buffer if, in the course of address computation, it gets updated to
   a. 0212h
   b. 01FCh

Buffer Length= (EAR-SAR+1)= 020F-0200+1=10h

   a. New Address Pointer = Updated Pointer - buffer length = 0212-10=0202h
   b. New Address Pointer = Updated Pointer + buffer length = 01FC+10=020Ch

12. Repeat the previous problem for SAR= 0210h and EAR=0201h

Buffer Length= (SAR-EAR+1)= 0210-0201+1=10h

   c. New Address Pointer = Updated Pointer - buffer length = 0212-10=0202h
   d. New Address Pointer = Updated Pointer + buffer length = 01FC+10=020Ch

2.7.2 Bit Reversed Addressing Mode

To implement FFT algorithms we need to access the data in a bit reversed manner. Hence a special addressing mode called bit reversed addressing mode is used to calculate the index of the next data to be fetched.

It works as follows. Start with index 0. The present index can be calculated by adding half the FFT length to the previous index in a bit reversed manner, carry being propagated from MSB to LSB.

Current index= Previous index + B (1/2(FFT Size))

13. Compute the indices for an 8-point FFT using Bit reversed Addressing Mode

Start with index 0. Therefore the first index would be (000)
Next index can be calculated by adding half the FFT length, in this case it is (100) to the previous index.
i.e. Present Index= (000)+B (100)= (100)
Similarly the next index can be calculated as

Present Index= (100) + B (100) = (010)

The process continues till all the indices are calculated. The following table summarizes the calculation.

<table>
<thead>
<tr>
<th>Index in Binary</th>
<th>BCD value</th>
<th>Bit reversed index</th>
<th>BCD value</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>010</td>
<td>2</td>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>011</td>
<td>3</td>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>101</td>
<td>5</td>
<td>101</td>
<td>5</td>
</tr>
<tr>
<td>110</td>
<td>6</td>
<td>011</td>
<td>3</td>
</tr>
<tr>
<td>111</td>
<td>7</td>
<td>111</td>
<td>7</td>
</tr>
</tbody>
</table>

2.8 Address Generation Unit

The main job of the Address Generation Unit is to generate the address of the operands required to carry out the operation. They have to work fast in order to satisfy the timing constraints.
As the address generation unit has to perform some mathematical operations in order to calculate the operand address, it is provided with a separate ALU.

Address generation typically involves one of the following operations.

a. Getting value from immediate operand, register or a memory location
b. Incrementing/ decrementing the current address
c. Adding/subtracting the offset from the current address
d. Adding/subtracting the offset from the current address and generating new address according to circular addressing mode
e. Generating new address using bit reversed addressing mode

The block diagram of a typical address generation unit is as shown in figure 2.13.
2.9 Programmability and Program Control

A programmable DSP device should provide the programming capability involving branching, looping and subroutines.

The implementation of repeat capability should be hardware based so that it can be programmed with minimal or zero overhead. A dedicated register can be used as a counter.

In a normal subroutine call, return address has to be stored in a stack thus requiring memory access for storing and retrieving the return address, which in turn reduces the speed of operation. Hence a LIFO memory can be directly interfaced with the program counter.

2.9.1 Program Control

Like microprocessors, DSP also requires a control unit to provide necessary control and timing signals for the proper execution of the instructions. In microprocessors, the controlling is micro coded based where each instruction is divided into microinstructions stored in micro memory. As this mechanism is slower, it is not applicable for DSP applications. Hence in DSP the controlling is hardwired base where the Control unit is designed as a single, comprehensive, hardware unit. Although it is more complex it is faster.

2.9.2 Program Sequencer

It is a part of the control unit used to generate instruction addresses in sequence needed to access instructions. It calculates the address of the next instruction to be fetched. The next address can be from one of the following sources.

a. Program Counter  
b. Instruction register in case of branching, looping and subroutine calls  
c. Interrupt Vector table  
d. Stack which holds the return address

The block diagram of a program sequencer is as shown in figure 2.14.
Program sequencer should have the following circuitry

a. PC has to be updated after every fetch

b. Counter to hold count in case of looping

c. A logic block to check conditions for conditional jump instructions

d. Condition logic-status flag