The Central Arithmetic Logic Unit

EXERCISE OBJECTIVES

Upon completion of this exercise, you will be familiar with the role that the CALU plays within a DSP.

DISCUSSION

Note: Some 'C50 assembler CALU instructions are briefly covered in this exercise. It will be left up to you, the student, to cover the rest of the related material. The material can be found in the following file: C:\LV91027\DOC\TMS320C5x_UsersGuide.pdf.

The Central Arithmetic Logic Unit (CALU) is where the most important signal processing manipulations take place.

The CALU, also known as the data path, is the principle arithmetic and logic processing path for a DSP.

It lies along the data (operand) bus and is an integral part of the execution of nearly every instructions.
The Central Arithmetic Logic Unit

A fixed-point CALU contains:

- Multiplier(s)
- Accumulator(s)
- Operand registers
- Shifters
- At least one Arithmetic Logic Unit (ALU)

Signal processing algorithms are almost entirely devoted to arithmetic and logic operations. The CALU is designed to execute these types of operations extremely rapidly.

A DSP is differentiated from a general-purpose processor by:

1. Its memory architecture (a DSP usually has a Harvard architecture).
2. The rapid execution time of the CALU (or data path).
The Central Arithmetic Logic Unit

Both the Multiplier and the ALU are simultaneously used during a MAC instruction. The CALU is said to be using its entire computational bandwidth.

For most DSPs, when the entire computational bandwidth of the CALU is repetitively used, a result is produced every clock cycle.

The operand registers play an important role within the CALU.

The registers are used to temporarily store operands, before they are supplied for arithmetic operations to the ALU or Multiplier.

The CALU of the TMS320C50 (‘C50) has 3 operand registers.

Memory-mapped Temporary REGister 0 (TREG0) is an operand register used by the Multiplier.

It holds one of the multiplication operands for the Multiplier.

The Product REGister (PREG) is a 32-bit operand register which stores the Multiplier result.

The value held in the PREG can be sent to the ALU for an arithmetic operation, or it can be passed on to the Data Bus (DB) for the another stage of processing.

ACCB (the ACCumulator Buffer) provides a temporary storage place for the value held by in the ACCumulator register (ACC).

The ACC register is designed to hold the last arithmetic result produced by the ALU.

The ALU is designed to implement a wide range of arithmetic and logical operations.
The Central Arithmetic Logic Unit

<table>
<thead>
<tr>
<th>EXAMPLE</th>
<th>OPERAND 1</th>
<th>OPERAND 2</th>
<th>OPERATION</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1011 0100</td>
<td>0001 1101</td>
<td>ADD</td>
<td>1101 0001</td>
</tr>
<tr>
<td>2</td>
<td>1011 0100</td>
<td>0001 1010</td>
<td>SUBTRACT</td>
<td>1001 1010</td>
</tr>
<tr>
<td>3</td>
<td>0010 1001</td>
<td>1011 1101</td>
<td>AND</td>
<td>0010 1001</td>
</tr>
<tr>
<td>4</td>
<td>0010 1001</td>
<td>1011 1101</td>
<td>OR</td>
<td>1011 1101</td>
</tr>
<tr>
<td>5</td>
<td>0111 0101</td>
<td>-</td>
<td>NEGATE</td>
<td>10001011</td>
</tr>
</tbody>
</table>

Some operations that are commonly executed by the ALU include: addition, subtraction, negation, and logical and, or, xor, and not.

The majority of ALU instructions execute within a single clock cycle.

Most of the ALU instructions that take more than one clock cycle rely on other units for pre- or post-processing of data.

E.g., add a data value to the ACC and then execute a binary shift. The TMS320C50 requires 2 clock cycles to execute the operation. The binary shift is an example of the type of processing that takes place after addition.

The ALUs of fixed-point DSPs execute 2s-complement arithmetic.

The ALU executes operations using twice the precision of the native word width of the processor.

For example the ALU of the ‘C50, a 16-bit fixed-point DSP, inputs, outputs, and executes with a 32-bit word width.
Most DSP have an ALU mode of operation called \textit{sign-extension} mode.

When enabled all ALU outputs are sign-extended.

Sign extension prevents a negative number from being mistaken for a positive one.

When the number of bits used to represent a word (e.g., 16 bits) is less than the number of bits required to represent the same word inside of the CALU (32 bits) then sign-extension extends the sign-bit into the added MSBs.

If the following 16-bit 2s-format number:

\begin{verbatim}
1011 0111 0010 0001 b
\end{verbatim}

was loaded using the ALU into a 32-bit Accumulator when sign-extension mode was enabled what would be the contents of the Accumulator register?

\begin{itemize}
  \item[a.] 0000 0000 0000 0000 1011 0111 0010 0001 b
  \item[b.] 1011 0111 0010 0001 b
  \item[c.] 1111 1111 1111 1110 1011 0111 0010 0001 b
  \item[d.] None of the above.
\end{itemize}
The last arithmetic or logical operation executed by the ALU is stored in the ACCumulator (ACC).

The result held in the ACC can either be stored in the ACC Buffer register (ACCB), passed on to the ALU, or to another stage of processing using the Data Bus (DB).

In the case of the 'C50 DSP, two operands need to be input into the ALU to execute any of its arithmetic or logical operations.

One of the operands is supplied by the ACCumulator register (ACC).

One of three other locations provide the other data operand for an ALU operation:

- Data path (e.g., to fetch an operand from memory)
- Multiplier Product REGister (PREG)
- ACCumulator Buffer (ACCB) register

Multiplication is an essential operation used in virtually all digital signal processing applications.

In many of the applications where multiplication is used half or more of the instructions executed by the processor are multiplication operations.

Central to nearly all programmable digital signal processors is the single-cycle Multiplier.

The Multiplier refers to the circuit within the DSP that executes the multiplication of binary numbers.

Depending on operand size (8-bit or 16-bit for the C50), nearly all Multiplier instructions can be executed within one clock cycle.
The Central Arithmetic Logic Unit

Multiplication in fixed-point DSPs is executed with 2s-complement arithmetic.

A Multiplier requires a minimum two operands to execute a multiplication.

These operands are treated as 2s-complement numbers.

In the TMS320C50, register TREG0 is always used as one of the operand sources for the Multiplier.

In certain cases, such as when the square root instructions (SQRA and SQRS) are executed, there are no other operands than TREGO used by the Multiplier.

When another multiplication operand is required it is fetched from one of two other locations:

- Data memory using the Data Bus (DB)
- Program memory using the Program Bus (PB)
As previously stated, the Multiplier result is stored in a Product REGister (PREG).

The product register is twice as wide as the word width of the multiplication operands (native data word width of the DSP).

<table>
<thead>
<tr>
<th>OPERAND 1</th>
<th>OPERAND 2</th>
<th>OPERATION</th>
<th>RESULT</th>
<th>PREG (AFTER SIGN EXT.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0111 0111</td>
<td>0011 0111</td>
<td>MULTIPLIER</td>
<td>0001 1001 1001 0001</td>
<td>0001 1001 1001 0001</td>
</tr>
<tr>
<td>(+ 119)</td>
<td>(+ 55)</td>
<td></td>
<td>(+ 6545)</td>
<td>(+ 6545)</td>
</tr>
<tr>
<td>0110 0110</td>
<td>1011 0111</td>
<td>MULTIPLIER</td>
<td>0010 0010 1110 1010</td>
<td>1110 0010 1110 1010</td>
</tr>
<tr>
<td>(+ 102)</td>
<td>(- 73)</td>
<td></td>
<td>(+ 8996)</td>
<td>(- 7446)</td>
</tr>
</tbody>
</table>

All Multiplier results are sign-extended before they are stored in the Product REGister (PREG).

This combined with the fact that the PREG has twice the operand word width means that, by itself, the Multiplier does not introduce any errors into computations.

To keep the level of arithmetic precision constant, the number of bits that are used to represent multiplication, accumulation and other arithmetic operation results, need to be increased.

That is why that in DSPs the Multiplier Product Register and the ALU ACCcumulator (ACC) have a width twice that of the native data word width.
The Central Arithmetic Logic Unit

<table>
<thead>
<tr>
<th>OPERAND 1</th>
<th>OPERAND 2</th>
<th>OPERATION</th>
<th>ACCUMULATOR</th>
<th>OVM CORRECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>OVER-FLOW</td>
<td>7FFF FFFF h</td>
<td>7FFF FFFF h</td>
<td>ADDITION</td>
<td>FFFF FFFE h</td>
</tr>
<tr>
<td>UNDER-FLOW</td>
<td>8000 0000 h</td>
<td>8000 0000 h</td>
<td>ADDITION</td>
<td>0000 0000 h</td>
</tr>
</tbody>
</table>

Maximum positive value: 7FFF FFFF h, $2^{31} - 1$

Maximum negative value: 8000 0000 h, $-2^{31}$

An overflowed value: FFFF FFFF h, $-1$

An underflowed value: 0000 0000 h, 0

Most signal processing applications require the addition of series of data values. These operations when executed within fixed-point DSP can easily lead to an overflow or underflow.

In many processors, a mode of operation exists which is used to decrease the error that is caused when overflow or underflow occurs. This mode within the TMS320C50 DSP is named Overflow saturation Mode (OVM).

Which of the following operations produce overflow of a 32-bit accumulator?

a. $4000 0019 h + 3333 ABB4 h$
b. $3B56 FF5F h + 5432 1145 h$
c. $(0455 E089 h + 0054 31AB h)$
d. $(1223 556F h + 2000 EF02 h)$

Barring the occurrence of overflow or underflow the precision level within the ALU and the Multiplier is kept at the same level as when the arithmetic entered the CALU.

However, at some point it is usually necessary to reduce the precision of the results; The data bus is still only half the bit-width of the CALU results.

Therefore, the programmer must select the product register or accumulator bits which will be passed on to the next stage of processing (via the data bus).
The selection of which bits to pass on is done with shifters that are located at the exit of the PREG and of the ACC.

A shifter can shift a binary number to the right or to the left by so many bits.

However, shifting a number \( n \) bits to the left effectively multiplies it by a power of two \( (2^n) \).

Pre- and Postscalers are used to scale values before they are input to or output from the Multiplier and ALU.

Scaling is an important operation in fixed-point DSPs because overflow can be avoided by prescaling CALU inputs.
The Central Arithmetic Logic Unit

<table>
<thead>
<tr>
<th>DSP FAMILY</th>
<th>METHOD USED TO AVOID OVERFLOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT&amp;T DSP16xx</td>
<td>4 guard bits</td>
</tr>
<tr>
<td>Analog Devices ADSP-21xx</td>
<td>8 guard bits</td>
</tr>
<tr>
<td>TI TMS320C2x and C5x</td>
<td>No guard bits.</td>
</tr>
<tr>
<td></td>
<td>Intermediate results can be scaled.</td>
</tr>
</tbody>
</table>

Ideally, the size of an accumulator register should be larger than the size of the multiplier product register by several bits.

The extra bits named guard bits allow the programmer to accumulate a number of values without the risk of overflowing the accumulator and without the need to scale intermediate results (avoiding overflow).

A single-bit field, present in the ‘C50, and known as the carry bit (or the C bit), is associated with the ACC register.

The C bit indicates whether an ALU operation generated a carry or a borrow.

The DSP can be programmed to conditionally test this bit.

The C bit, similar to a guard bit, is useful for extended-precision arithmetic.
The Central Arithmetic Logic Unit

PROCEDURE

The ALU

In this procedure section, you will load the accumulator with a value input from the DIP switch. Then, using the ALU, you will add three different values to the accumulator, each one fetched from a different operand source.

Note: Before using the C5x VDE please make certain the circuit board power source is turned ON, and that the serial connection is present between the host computer and the DIGITAL SIGNAL PROCESSOR circuit block labeled SERIAL PORT.

If at anytime during the following procedure you realize that you did not correctly follow a procedure step, then using the C5x VDE simply edit the PC (Program Counter register), back to one of the last labeled program memory addresses (either MAIN, MARKER1, MARKER2, or MARKER3).

Within WinFACET, using the Go to previous page button, return to the beginning of the Procedure Section associated with the labeled program address that you returned to and start following the procedure steps once again.

☐ 1. Open the ex2_1.asm assembler source file within an ASCII text editor. This is the DSP program source file used for the exercise. You can refer to this source file at anytime during the procedure.

☐ 2. Open the C5x VDE, and load the ex2_1.dsk DSP program file.

☐ 3. Using the C5x VDE, open a Data Memory display to 0980h. This is the data memory address where the constants and variables used by the program begin being stored.
The Central Arithmetic Logic Unit

4. Using the C5x VDE, open a Program Memory display to 0A0Dh. This is the program memory address where program machine code was stored.

5. Within the Dis-Assembly window, place a breakpoint at the program address labeled MAIN and then press the C5x VDE RUN command, this will initialize the CPU registers required for the exercise.

6. Using the C5x VDE, edit the contents of the PREG and ACCB registers. Input different 16-bit 2s-complement values (0000 XXXXh), of your choosing, into the registers. They must be 16-bit values.

7. Position the eight on-off switches (DIP switch) located in the I/O INTERFACE circuit block, found on the DIGITAL SIGNAL PROCESSOR board, to a value of your choosing. This value will be loaded into the accumulator register, ACC.

8. Using the C5x VDE, STEP OVER the CALL DIPSWITCH:

   CALL A80h,* instruction.

   Observe that the binary value that the DIP switch was set to has been loaded into the accumulator register (ACC).

   What is the content of the data memory address labeled INPUT?
   a. 0010 1111 1011 0011b
   b. 01FEh
   c. The same value that you input through the DIP switch.
   d. None of the above

9. Using the C5x VDE, STEP OVER the ADD, APAC and ADDB instructions.

   The following mathematical operation was executed by the above instructions,
   
   \[ \text{ACC} = \text{ACC} + \text{VALUE} + \text{PREG} + \text{ACCB} \]

10. Write down the value contained in the ACC.
The Central Arithmetic Logic Unit

Sign-Extension Mode inside of the ALU

In this procedure section, you will enable sign-extension mode, execute the code of the previous procedure section, and compare the results generated by the ALU for both procedure sections.

☐ 11. Using the C5x VDE, edit the SXM bit to 1.

   Sign-extension mode is enabled in the DSP.

☐ 12. Make certain that the PREG, ACCB and the DIP switch have the same values as you had chosen before, if not, then set them back to the same values.

☐ 13. Using the C5x VDE, edit the contents of the PC register to the program address labeled MAIN. The execution line will return to the instruction labeled MAIN.

☐ 14. Using the C5x VDE, execute, once again, with the STEP OVER command, the instruction: CALL A80h,* (and the three add instructions that follow).

☐ 15. Write down the value contained in the ACC.

☐ 16. Compare the first accumulator result that you wrote down (this one was generated with sign-extension mode not enabled) with the second accumulator result that you just wrote down (this one was generated with sign-extension mode enabled).

☐ 17. Observe that the results are not the same. This is because, when the SXM bit is enabled, the negative 16-bit wide value stored in the data address labeled VALUE, and added to the accumulator with the ADD ch instruction, is sign-extended and seen as a negative number by the ALU. This is contrary to when SXM is not enabled.

Prescaling and Postscaling inside of the ALU

In this procedure section, you will use the prescaler located at the input of the ALU.

☐ 18. Using the C5x VDE, make certain that the SXM bit is set (SXM = 1). If it is not already set then edit the SXM bit to 1.
The Central Arithmetic Logic Unit

19. Note the ADD instructions that follow the program address labeled MARKER1. They prescale an operand before adding it to the accumulator.

20. Using the C5x VDE, zero the contents of the ACC. STEP OVER the instruction: ADD #1111h,3

The value 1111h was added to the ACC register that previously contained zero. By observing the present contents of the ACC, which of the following choices describes what the prescaler did to the added value?

a. The added value was scaled by $2^3$.
b. The added value was scaled by $2^2$.
c. The added value was sign-extended.
d. The added value was scaled by $2^{-3}$.

21. Using the C5x VDE, again zero the contents of the ACC register. STEP OVER the instruction: ADD #8111h,15

The value 8111h (corresponding to the negative value -32495) was added to the ACC register that previously contained zero. By observing the present contents of the ACC, which of the following choices describes how the prescaler and ALU changed the added ACC value?

a. The added value was sign extended.
b. The added value was shifted 15 bits to the right and then sign-extended.
c. The added value was scaled by $2^{15}$ and sign-extended.
d. None of the above.

Both added values were taken from the 16-bit data bus and prescaled. The bus width between the prescaler and the ALU is 32 bits wide.

22. Using the C5x VDE, slowly STEP OVER (while watching the I/O INTERFACE display) the instruction:

```
CALL DISPLAYHIGH,*
CALL DISPLAYLOW,*
```

The value of the ACC will output to the 7 segments

23. Using the C5x VDE, STEP OVER the ZAP instruction. The ZAP instruction will zero the ACC and PREG registers.
The Central Arithmetic Logic Unit

The Multiplier: Basic Operations

In this procedure section, you will execute basic multiplication operations, and by enabling one of the product shift modes, shift the output of the product register.

<table>
<thead>
<tr>
<th>.ds</th>
<th>00980h</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPUT</td>
<td>.word 0</td>
</tr>
</tbody>
</table>

| X0   | .q14  | 0.48 |
| X1   | .q14  | 0.95 |
| X2   | .q14  | 0.35 |
| X3   | .q14  | 1.00 |
| B0   | .q14  | 0.22 |
| B1   | .q14  | 0.64 |
| B2   | .q14  | 0.95 |
| B3   | .q14  | 1.11 |

There are eight data values used in this procedure section. Each value is written as a Q14-format binary number.

The data values are stored in the dmas labeled X0 to X3 and B0 to B3.

☐ 24. Using the C5x VDE, make certain that the following is true:

- SXM = 1
- ACC = 0000 0000h
- PREG = 0000 0000h
- PM = 0

☐ 25. Using the C5x VDE, STEP OVER the LT instruction.

This loads the content of the data memory address labeled X0 into TREG0. TREG0 is an operand source for the multiplier.

Are the contents of the dma labeled X0 and of the TREG0 register the same?

☐ Yes   ☐ No

☐ 26. Using the C5x VDE, STEP OVER the following MPY instruction.

The contents of the data memory address labeled B0 (the fifth data value) are multiplied with the contents of TREG0.
Observe, using the C5x VDE, that the PREG holds the product of the contents of the data memory address labeled B0 with TREG0.

☐ 27. Using the C5x VDE, STEP OVER the instruction: APAC.

The APAC instruction adds the PREG to the ACC.

Observe that the contents of the accumulator register (ACC), and of the product register (PREG) are the same, both are equal to 01B0 7660h. ACC was equal to 0000h before executing the APAC instruction.

☐ 28. Using the C5x VDE, edit the PC register to return the execution line to the program address labeled MARKER2. Edit the content of the PM bits to 1 (this enables the product-shifter, the PREG output is left shifted by 1-bit).

☐ 29. Using the C5x VDE, once again zero the contents of the ACC register. STEP OVER the LT, MPY and APAC instructions executed in steps 25 to 27.

Notice the effect that the postscaler has on the contents of the accumulator, the PREG was shifted left by 1-bit (multiplied by 2).

If the values entered into the multiplier were written in Q14-format and if a product-shift of 1-bit to the left occurred, then what would be the numerical format of the value contained in the accumulator register?

a. Q13-format
b. Q14-format
c. Q28-format
d. Q27-format
The Central Arithmetic Logic Unit

The Multiplier: Overflow and Overflow Saturation Mode

In this procedure section, you will make the accumulator overflow and then you will enable OVerflow saturation Mode (OVM) to protect against it occurring again.

☐ 30. Using the C5x VDE, make certain that the following is true:

\[
\begin{align*}
\text{ACC} &= 0000\ 0000h \\
\text{PREG} &= 0000\ 0000h \\
\text{TREG0} &= 0000h \ PM = 0 \\
\text{OV} &= 0 \\
\text{SXM} &= 1 \\
\text{OVM} &= 0
\end{align*}
\]

☐ 31. Using the C5x VDE, edit the Program Counter register, PC, to the program address labeled MARKER2.

☐ 32. Using the C5x VDE, STEP OVER the instructions located between the program address labeled MARKER2 and the instruction:

\[
\text{B \ MARKER2,*}
\]

☐ 33. Execute the, \text{B \ MARKER2,*} instruction. This will branch the execution line back to the program address labeled MARKER2 (this has the same effect as editing PC).

☐ 34. Using the STEP OVER command, continue executing the LT, MPY, APAC, and \text{B \ MARKER2} instructions, until OV bit is equal to 1.
The Central Arithmetic Logic Unit

While executing these instructions, observe that the value held in the ACC register is becoming larger. ACC overflow occurs when \( OV = 1 \).

What is the value of the ACC?

a. 44FD 8DC8h
b. 89FB 1B90h
c. 7838 9B90h
d. 8BAB 91F0h

35. Using the C5x VDE, edit the PC register to the program address labeled MARKER2. Zero the OV bit and the ACC, PREG, TREG0 registers.

36. Set the OVM bit to 1.

This enables Overflow saturation Mode (OVM) in the DSP.

37. Once again, STEP OVER the LT, MPY, APAC, and B MARKER2 instructions until the OV bit is equal to 1.
The Central Arithmetic Logic Unit

When the accumulator overflow occurred, and OVerflow saturation Mode (OVM) was not enabled, the result contained in the ACC had a relative error (Δ) of 186% compared with the correct value.

However, when OVM was enabled and the same overflow occurred the result contained in the ACC only had a relative error (Δ) of 7% compared with the correct value.

Multiplier Postscaling

In this procedure section, you will add 128 very large values together, consecutively, and prevent an accumulator overflow by setting the Product-shift Mode (PM) so that the output of the PREG is scaled by $2^6$.

☐ 38. Using the C5x VDE, edit the PC to the program address labeled MARKER3. Clear the OV, and OVM bits.

☐ 39. Using the C5x VDE, STEP OVER the instruction: ZAP

This zeroes the accumulator and product registers.

☐ 40. STEP OVER the instruction: SPM  3

This sets the Product-shift Mode (PM) bits to 3.

By setting the PM bits to 3, the output of the PREG will be shifted 6 bits to the right, which is equivalent to dividing it by $2^6$. 

![Diagram of correct values and overflow values]
41. Place a breakpoint at the program address labeled END_BLOCK.

The maximum positive value that can be represented in the 16-bit 2s-
format, 7FFFh, is used as the operand for the LT and MPY instructions. 
These instructions, located between the program address labeled 
MARKER4 and the one labeled END_BLOCK, fetch the contents of the 
data memory address labeled BIG_VALUE.

42. Using the C5x VDE, execute the RUN command.

The program is halted at the breakpoint, observe that after executing 128 
additions the accumulator register still has not overflowed (OV is still equal 
to 0).

43. Edit the PC to the program address labeled MARKER4. STEP OVER, once 
again, the LT, MPY, and APAC instructions (the 129th consecutive multiply- 
accumulate).

Observe that the accumulator overflows this time. Implying that when the 
output of the product register is scaled by $2^6$, a minimum of 128 
consecutive additions (of $7FFFh \times 7FFFh$) can be executed without 
causing overflow.

44. End the C5x VDE session.
CONCLUSION

- Fixed-point multipliers and ALUs execute 2’s-complement arithmetic.
- Multiplier results are sign-extended before they are stored in the product register. Sign extension prevents a negative number from being mistaken for a positive one.
- To keep the level of arithmetic precision constant within fixed-point DSPs, the product and accumulator registers are, at least, twice the native word width of the internal bus.
- In fixed-point DSPs, scaling is used to lower the risk of overflow and underflow from occurring and to select subsets of the CALU output bits.
- Overflow saturation mode is used to decrease the error that is caused when overflow or underflow occurs.

REVIEW QUESTIONS

1. Which of the following operand sources is always used by the ALU?
   a. The accumulator register (ACC).
   b. The product register (PREG).
   c. The accumulator buffer register (ACCB).
   d. A data memory address from the data bus.

2. What is the difference between using the DSP when sign-extension mode is enabled and when it is disabled?
   a. When enabled, all data values in the DSP are sign extended.
   b. When enabled, the accumulator saturates to the most positive or negative values when overflow or underflow occurs.
   c. When enabled, the multiplier output is sign extended.
   d. When enabled, the ALU output is sign extended.

3. Why, within the TMS320C50 DSP, are the accumulator and product registers twice the bit-width (32 bits) of the internal buses (16 bits)?
   a. To keep the level of arithmetic precision constant.
   b. To avoid overflow or underflow from occurring.
   c. All of the above.
   d. None of the above.
The Central Arithmetic Logic Unit

4. Which of the following elements is not part of the Central Arithmetic Logic Unit (CALU)?
   a. Auxiliary Register Arithmetic Unit (ARAU)
   b. Operand Registers
   c. Multiplier
   d. Arithmetic Logic Unit (ALU)

5. Which of the following choices is not used as a way of avoiding accumulator overflow in a DSP?
   a. Guard bits, extra bits in the accumulator.
   b. Product shifter
   c. Sign-extension mode
   d. None of the above.