Official COD Assignment I Solutions

1.

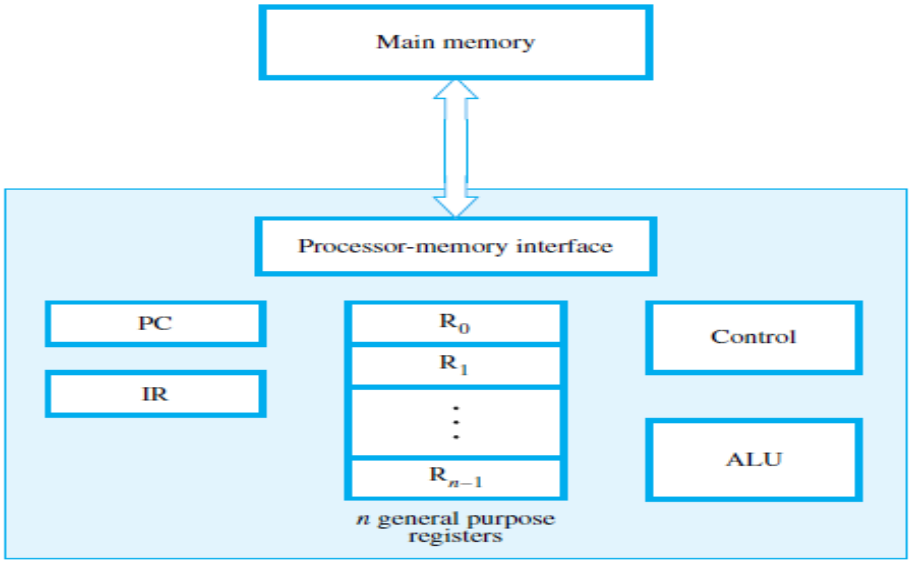


Figure 1.2 Connection between the processor and the main memory.

The *instruction register* (IR) holds the instruction that is currently being executed. Its output is available to the control circuits, which generate the timing signals that control the various processing elements involved in executing the instruction.

The *program counter* (PC) is another specialized register. It contains the memory address of the next instruction to be fetched and executed. During the execution of an instruction, the contents of the PC are updated to correspond to the address of the next instruction to be executed. It is customary to say that the PC *points* to the next instruction that is to be fetched from the memory. In addition to the IR and PC, *general-purpose registers* R0 through R*n*–1, often called processor registers. They serve a variety of functions, including holding operands that have been loaded from the memory for processing.

The processor-memory interface is a circuit which manages the transfer of data between the main memory and the processor.

If a word is to be read from the memory, the interface sends the address of that word to the memory along with a Read control signal. The interface waits for the word to be retrieved, then transfers it to the appropriate processor register. If a word is to be written into memory, the interface transfers both the address and the word to the memory along with a Write control signal.

The required steps are:

- Send the address of the instruction word from register PC to the memory and issue a Read control command.
- Wait until the requested word has been retrieved from the memory, then load it into register IR, where it is interpreted (decoded) by the control circuitry to determine the operation to be performed.
- Increment the contents of register PC to point to the next instruction in memory.
- Send the address value LOC from the instruction in register IR to the memory and issue a Read control command.
- Wait until the requested word has been retrieved from the memory, then load it into register R2.

2.

B	Values represented		
*b*2*b*1*b*0	Sign and magnitude	1’s complement	2’s complement
0 1 1	+3	+3	+3
0 1 0	+2	+2	+2
0 0 1	+1	+1	+1
0 0 0	+0	+0	+0
1 0 0	-0	-3	-4
1 0 1	-1	-2	-3
1 1 0	-2	-1	-2
1 1 1	-3	-0	-1

Decimal	Sign/Mag	1's	2’s
5	0000101	0000101	0000101
-2	1000010	1111101	1111110
26	0011010	0011010	0011010
-10	1001010	1110101	1110110
-19	1010011	1101100	1101101
51	0110011	0110011	0110011

3. To represent a value given in a certain number of bits by using a larger number of bits.

For a positive number, this is achieved by adding 0s to the left.

For a negative number in 2's-complement representation, the leftmost bit, which indicates the sign of the number, is a 1.

A longer number with the same value is obtained by replicating the sign bit to the left as many times as needed.

In summary, to represent a signed number in 2's-complement form using a larger number of bits, repeat the sign bit as many times as needed to the left. This operation is called *sign extension*.

Give example

Big-endian scheme:
Byte contents in hex, starting at location 1000, will be ASCII of A, B, C, D, E, F, G. The two words at 1000 and 1004 will be 41424344 and 454647XX. Byte 1007 (shown as XX) is unchanged.

Little endian scheme:
Byte contents in hex, starting at location 1000, will be 4A, 6F, 68, 6E, 73, 6F, 6E. The two words at 1000 and 1004 will be 44434241 and XX474645. Byte 1007 (shown as XX) is unchanged.

4. (i) (a) $7_{10} = 00111_2$ and $13_{10} = 01101_2$

Adding these two positive numbers, we obtain 10100, which is a negative number. Therefore, overflow has occurred.

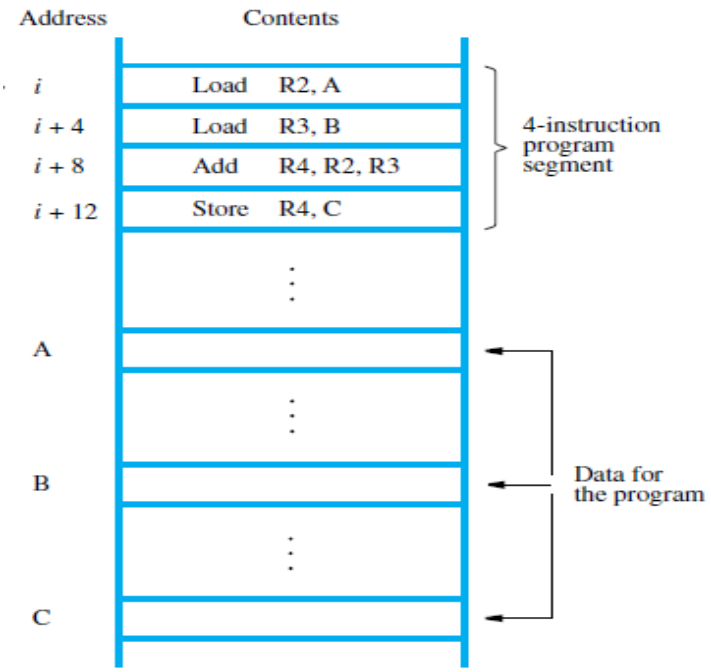
To subtract them, we first form the 2's-complement of 01101, which is 10011. Then we perform addition with 00111 to obtain 11010, which is -6_{10}, the correct answer.

(*b*) $-12_{10} = 10100_2$ and $9_{10} = 01001_2$

Adding these two numbers, we obtain 11101 = -3_{10}, the correct answer. To subtract them, we first form the 2's-complement of 01001, which is 10111. Then we perform addition of the two negative numbers 10100 and 10111 to obtain 01011, which is a positive number. Therefore, overflow has occurred.

(ii)

```
Load R2, A
Load R3, B
Add R4, R2, R3
Store R4, C
```



5.

$$A = 1.011011 \times 2^2$$
and
$$B = -1.101010 \times 2^0$$

The required operations are performed as follows:

Subtraction

According to the Add/Subtract rule we perform the following four steps:

1. Shift the mantissa of *B* to the right by two bit positions, giving 0.01101010.

2. Set the exponent of the result to 10001.

3. Subtract the mantissa of *B* from the mantissa of *A* by adding mantissas, because *B* is negative, giving

$$\begin{array}{r} 1.0\,1\,1\,0\,1\,1\,0\,0 \\ +\,0.0\,1\,1\,0\,1\,0\,1\,0 \\ \hline 1.1\,1\,0\,1\,0\,1\,1\,0 \end{array}$$

and set the sign of the result to 0 (positive).

4. The result is in normalized form, but the fractional part of the mantissa needs to be truncated to six bits. If this is done by rounding, the two bits to be removed represent the tie case, so we round to the nearest even number by adding 1, obtaining a result mantissa of 1.110110. The answer is

$A - B =$

0	10001	110110

Multiplication

According to the Multiplication rule, we perform the following three steps:

1. Add the exponents and subtract 15 to obtain 10001 as the exponent of the result.

2. Multiply mantissas to obtain 10.010110101110 as the mantissa of the result. The sign of the result is set to 1 (negative).

3. Normalize the resulting mantissa by shifting it to the right by one bit position. Then add 1 to the exponent to obtain 10010 as the exponent of the result. Truncate the mantissa fraction to six bits by rounding to obtain the answer

$A \times B =$

0	10010	001011