3.1 What is 5ED4 - 07A4 when these values represent unsigned 16-bit hexadecimal numbers? The result should be written in hexadecimal. Show your work.

Solution:

Unsigned 16-bit integers

0101 1110 1101 0100 0x5ED4

- 0000 0111 1010 0100 0x07A4

= 0101 0111 0011 0000 0x5730

3.6 Assume 185 and 122 are unsigned 8-bit decimal integers. Calculate 185 – 122. Is there overflow, underflow, or neither?

Solution:

Unsigned 8-bit integers

185 = 11 \* 16 + 9 => 0xB9

122 = 7 \* 16 + 10 => 0x7A

1011 1001 0xB9 (185 = 11\*16 + 9)

- 0111 1010 0x7A (122 = 7\*16 + 10)

= 0011 1111 0x3F (63 = 3\*16 + 15)

There is neither overflow nor underflow. ((+ui) – (+ui))

3.13 Using a table similar to that shown in Figure 3.6, calculate the

product of the hexadecimal unsigned 8-bit integers 62 and 12 using the hardware

described in Figure 3.5. You should show the contents of each register on each step.

Solution:

Multiplicand = 0x62 = 0110 0010

Multiplier = 0x12 = 0001 0010

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteration | Step | Multiplicand(8-bit)  (constant) | Product(16-bit) | |
| HI(8-bit) | LO(8-bit) |
| 0 | Init: HI = 0x00, LO = Multiplier | 0110 0010 | 0000 0000 | 0001 0010 |
| 1 | 1: 0 => No operation | 0110 0010 | 0000 0000 | 0001 0010 |
| 2: Shift right Product | 0110 0010 | 0000 0000 | 0000 1001 |
| 2 | 1: 1 => HI = HI + Multiplicand | 0110 0010 | 0110 0010 | 0000 1001 |
| 2: Shift right Product | 0110 0010 | 0011 0001 | 0000 0100 |
| 3 | 1: 0 => No operation | 0110 0010 | 0011 0001 | 0000 0100 |
| 2: Shift right Product | 0110 0010 | 0001 1000 | 1000 0010 |
| 4 | 1: 0 => No operation | 0110 0010 | 0001 1000 | 1000 0010 |
| 2: Shift right Product | 0110 0010 | 0000 1100 | 0100 0001 |
| 5 | 1: 1 => HI = HI + Multiplicand | 0110 0010 | 0110 1110 | 0100 0001 |
| 2: Shift right Product | 0110 0010 | 0011 0111 | 0010 0000 |
| 6 | 1: 0 => No operation | 0110 0010 | 0011 0111 | 0010 0000 |
| 2: Shift right Product | 0110 0010 | 0001 1011 | 1001 0000 |
| 7 | 1: 0 => No operation | 0110 0010 | 0001 1011 | 1001 0000 |
| 2: Shift right Product | 0110 0010 | 0000 1101 | 1100 1000 |
| 8 | 1: 0 => No operation | 0110 0010 | 0000 1101 | 1100 1000 |
| 2: Shift right Product | 0110 0010 | 0000 0110 | 1110 0100 |

Product = 0000 0110 1110 0100 = 0x06E4(HI = 0x06, LO = 0xE4)

P.S. Above table use a 8-bit multiplier. However, Figure 3.5 use a 32-bit hardware. I think it is unnecessary to show a 32-bit multiplication with two small number of 0x62 and 0x12. But if we must do in a 32-bit multiplier, we just add extra 24 0’s to the head of each register and then do 24 meaningless rounds after round 8. A little difference will occur during the shift right process and the result in HI/Lo registers. Below is the first several rounds.

Multiplicand = 0x62 = 0x00000062 = 0000 0000 0000 0000 000 0000 0110 0010

Multiplier = 0x12 = 0x00000012 = 0000 0000 0000 0000 000 0000 0001 0010

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteration | Step | Multiplicand(32-bit)  (constant) | Product(64-bit) | |
| HI(32-bit) | LO(32-bit) |
| 0 | Init: HI = 0x00, LO = Multiplier | 0000 0000 0000 0000 0000 0000 0110 0010 | 0000 0000 0000 0000 0000 0000 0000 0000 | 0000 0000 0000 0000 0000 0000 0001 0010 |
| 1 | 1: 0 => No operation | 0000 0000 0000 0000 0000 0000 0110 0010 | 0000 0000 0000 0000 0000 0000 0000 0000 | 0000 0000 0000 0000 0000 0000 0001 0010 |
| 2: Shift right Product | 0000 0000 0000 0000 0000 0000 0110 0010 | 0000 0000 0000 0000 0000 0000 0000 0000 | 0000 0000 0000 0000 0000 0000 0000 1001 |
| 2 | 1: 1 => HI = HI + Multiplicand | 0000 0000 0000 0000 0000 0000 0110 0010 | 0000 0000 0000 0000 0000 0000 0110 0010 | 0000 0000 0000 0000 0000 0000 0000 1001 |
| 2: Shift right Product | 0000 0000 0000 0000 0000 0000 0110 0010 | 0000 0000 0000 0000 0000 0000 0011 0001 | 0000 0000 0000 0000 0000 0000 0000 0100 |
| 3 | 1: 0 => No operation | 0000 0000 0000 0000 0000 0000 0110 0010 | 0000 0000 0000 0000 0000 0000 0011 0001 | 0000 0000 0000 0000 0000 0000 0000 0100 |
| 2: Shift right Product | 0000 0000 0000 0000 0000 0000 0110 0010 | 0000 0000 0000 0000 0000 0000 0001 1000 | 1000 0000 0000 0000 0000 0000 0000 0010 |
| 4 | 1: 0 => No operation | 0000 0000 0000 0000 0000 0000 0110 0010 | 0000 0000 0000 0000 0000 0000 0001 1000 | 1000 0000 0000 0000 0000 0000 0000 0010 |
|  | 2: Shift right Product | 0000 0000 0000 0000 0000 0000 0110 0010 | 0000 0000 0000 0000 0000 0000 0000 1100 | 0100 0000 0000 0000 0000 0000 0000 0001 |
| 5 | 1: 1 => HI = HI + Multiplicand | 0000 0000 0000 0000 0000 0000 0110 0010 | 0000 0000 0000 0000 0000 0000 0110 1110 | 0100 0000 0000 0000 0000 0000 0000 0001 |
|  | 2: Shift right Product | 0000 0000 0000 0000 0000 0000 0110 0010 | 0000 0000 0000 0000 0000 0000 0011 0111 | 0010 0000 0000 0000 0000 0000 0000 0000 |
| 6 | 1: 0 => No operation | 0000 0000 0000 0000 0000 0000 0110 0010 | 0000 0000 0000 0000 0000 0000 0011 0111 | 0010 0000 0000 0000 0000 0000 0000 0000 |
|  | 2: Shift right Product | 0000 0000 0000 0000 0000 0000 0110 0010 | 0000 0000 0000 0000 0000 0000 0001 1011 | 1001 0000 0000 0000 0000 0000 0000 0000 |
| …… | | | | |
| 32 | 1: 0 => No operation | 0000 0000 0000 0000 0000 0000 0110 0010 | 0000 0000 0000 0000 0000 0000 0000 0000 | 0000 0000 0000 0000 0000 1101 1100 1000 |
|  | 2: Shift right Product | 0000 0000 0000 0000 0000 0000 0110 0010 | 0000 0000 0000 0000 0000 0000 0000 0000 | 0000 0000 0000 0000 0000 0110 1110 0100 |

HI = 0000 0000 0000 0000 0000 0000 0000 0000 = 0x00000000

LO = 0000 0000 0000 0000 0000 0110 1110 0100 = 0x000006E4

Product = 0x00000000 000006E4