**TUTORIAL 1: Numbering Systems**

1. Convert the following to decimal numbers.
2. 111001112

|  |
| --- |
| **Answer** |
| 111001112 = (1x27)+(1 x26)+(1 x25)+(0 x24)+(0 x23)+(1 x22)+(1 x21)+(1 x20)  = 128 + 64 + 32 + 0 + 0 + 4 + 2 + 1  = **23110** |

1. 55778

|  |
| --- |
| **Answer** |
| 55778 = (5 x83)+(5 x82)+(7 x81)+(7 x80)  = 2560 + 320 +56 + 7  = **294310** |

1. ABCD H

|  |
| --- |
| **Answer** |
| ABCD H = (A x163)+(B x162)+(C x161)+(D x160)  = 40960 + 2816 + 192 + 13  = **4398110** |

1. Convert the followings to Base-2, Base-8 and Base-16 respectively.
2. 3064 D

|  |  |  |
| --- | --- | --- |
| **Answer** | | |
| Base-2 | Base-8 | Base-16 |
| 2) 3064 - 0  2) 1532 – 0  2) 766 – 0  2) 383 – 1  2) 191 – 1  2) 95 – 1  2) 47 – 1  2) 23 – 1  2) 11 – 1  2) 5 – 1  2) 2 – 0  1  Answer: **1011 1111 10002** | 8) 3064 – 0  8) 383 – 7  8) 47 – 7  5  Answer: **57708** | 16) 3064 – 8  16) 191 – F  B  Answer: **BF816** |

1. 201610

|  |  |  |
| --- | --- | --- |
| **Answer** | | |
| Base-2 | Base-8 | Base-16 |
| 2) 2016 - 0  2) 1008 – 0  2) 504 – 0  2) 252 – 0  2) 126 – 0  2) 63 – 1  2) 31 – 1  2) 15 – 1  2) 7 – 1  2) 3 – 1  1  Answer: **111 1110 00002** | 8) 2016 – 0  8) 252 – 4  8) 31 – 7  3  Answer: **37408** | 16) 2016– 0  16) 126 – E  7  Answer: **7E016** |

1. 2899 base-10

|  |  |  |
| --- | --- | --- |
| **Answer** | | |
| Base-2 | Base-8 | Base-16 |
| 2) 2899 – 1  2) 1449 – 1  2) 724 – 0  2) 362 – 0  2) 181 – 1  2) 90 – 0  2) 45 – 1  2) 22 – 0  2) 11 – 1  2) 5 – 1  2) 2 – 0  1  Answer: **1011 0101 00112** | 8) 2899 – 3  8) 362 – 2  8) 45 – 5  5  Answer: **55238** | 16) 2899 – 3  16) 181 – 5  B  Answer: **B5316** |

1. Convert the following numbers to the respective base.
2. 11111010001111002 to base-8 and base-16 respectively

|  |  |
| --- | --- |
| **Answer** | |
| Base-8 | Base-16 |
| 11111010001111002  = 1/111/101/000/111/100  = **1 7 5 0 7 48** | 11111010001111002  = 1111/1010/0011/1100  = **F A 3 C16** |

1. 11000011101001012 to base-8 and base-16 respectively

|  |  |
| --- | --- |
| **Answer** | |
| Base-8 | Base-16 |
| 11000011101001012  = 1/100/001/110/100/101  = **1 4 1 6 4 58** | 11000011101001012  = 1100/0011/1010/0101  = **C 3 A 516** |

1. 1EFH to base-2 and base-8 respectively

|  |  |
| --- | --- |
| **Answer** | |
| Base-2 | Base-8 |
| 1EFH  = **0001 1110 11112** | 1EFH  = 0001 1110 11112  = 000/111/101/111  = **0 7 5 7 8** |

1. 257ACH to base-2 and base-8 respectively

|  |  |
| --- | --- |
| **Answer** | |
| Base-2 | Base-8 |
| 257ACH  = **0010 0101 0111 1010 11002** | 257ACH  = 0010 0101 0111 1010 11002  = 00/100/101/011/110/101/100  = **0 4 5 3 6 5 4 8** |

1. 7548 to base-16

|  |
| --- |
| **Answer** |
| 7548 **= 111 101 1002**  = 1/1110/1100  = **1 E C16** |

1. 44716 to base-8

|  |
| --- |
| **Answer** |
| 44716 = 0100 0100 01112  = 010/001/000/111  = **2 1 0 78** |

1. Perform the following conversions. You are required to show the working steps clearly. If the operation(s) is illogical, explain the reason.

(a) 165810 to hexadecimal number (PYP-08/14: 2 marks)

|  |
| --- |
| **Answer** |
| 16) 1658 – A  16) 103 – 7  6  Answer = **67A16** |

(b) 7658 to hexadecimal number (PYP-08/14: 2 marks)

|  |
| --- |
| **Answer** |
| 7658 = 111 110 1012  = 1/1111/0101  = **1 F 516** |

(c) 673247 to decimal number (PYP-08/13: 2 marks)

|  |
| --- |
| **Answer** |
| 673247 = Illogical. Because base-7 does not include 7. |

(d) 8739010 to base-5 number (PYP-08/13: 2 marks)

|  |
| --- |
| **Answer** |
| 5)87390 – 0  5) 17478 – 3  5) 3495 – 0  5) 699 – 4  5) 139 – 4  5) 27 – 2  5) 5 – 0  1  Answer: **102440305** |

(e) 392088 to decimal number (PYP-04/13: 2 marks)

|  |
| --- |
| **Answer** |
| 392088 = Illogical. Because base-8 consists of numbers range from 0 to 7 only. |

1. Perform the following conversions. You are required to show the working steps clearly. If the operation(s) is illogical, explain the reason.

(a) 123010 to base-6 number (PYP-04/13: 2 marks)

|  |
| --- |
| **Answer** |
| 6) 1230 – 0  6) 205 – 1  6) 34 – 4  5  Answer: **54106** |

(b) 152810 to hexadecimal number (PYP-08/15: 2 marks)

|  |
| --- |
| **Answer** |
| 16) 1528 – 8  16) 95 – F  5  Answer: **5F816** |

(c) 62178 to decimal number (PYP-08/15: 2 marks)

|  |
| --- |
| **Answer** |
| 62718 = (6 x83)+(2 x82)+(1 x81)+(7 x80)  = 3072 + 128 + 8 + 7  = **321510** |

(d) 3728 to hexadecimal number (PYP-08/15: 2 marks)

|  |
| --- |
| **Answer** |
| 3728 = 011 111 0102  = 0/1111/1010  = **0 F A16** |

(e) 23010 to base-7 number (PYP-01/14: 2 marks)

|  |
| --- |
| **Answer** |
| 7) 230 – 6  7) 32 – 4  4  Answer : **4467** |

1. Perform the following operations and show the answers in the respective number base. You are required to show your working steps clearly

[NOTE: every 16, carry 1 to next]

1. ABC16 + FFF16 (PYP-01/14: 3 marks)

|  |
| --- |
| **Answer** |
| A B C H  + F F F H  **1 A B B16** |

1. 1258 – 778 (PYP-01/14: 2 marks)

|  |
| --- |
| **Answer** |
| 1 2 58  - 7 78  **2 68** |

1. 10112 x 1012

|  |
| --- |
| **Answer** |
| 1 0 1 12  X 1 0 12  1 0 1 1  1 0 1 1 0  **1 1 0 1 1 12** |

1. 110112 x 10112

|  |
| --- |
| **Answer** |
| 1 1 0 1 12  X 1 0 1 12  1 1 0 1 1  1 1 0 1 1  1 1 0 1 1 0  **1 0 0 1 0 1 0 0 12** |

1. 100111002 + 010010112

|  |
| --- |
| **Answer** |
| 1 0 0 1 1 1 0 02  + 0 1 0 0 1 0 1 12  **1 1 1 0 0 1 1 12** |

1. Perform the following operations and show the answers in the respective number base. You are required to show your working steps clearly
2. 4618 + 5158

|  |
| --- |
| **Answer** |
| 4 6 18  + 5 1 58  **1 1 7 68** |

1. 2247 - 1367

|  |
| --- |
| **Answer** |
| 2 2 47  - 1 3 67  **557** |

1. 173H x AAH

|  |
| --- |
| **Answer** |
| 1 7 3 H  X A A H  E 7 E  E 7 E  **F 6 5 E H** |

1. 111110102 + 11108 (Show your final answer in Hex format.) (PYP-04/13: 3 marks)

|  |
| --- |
| **Answer** |
| 111110102 = 1111/1010  = F A16  11108 = 001 001 001 0002  = 0010/0100/1000  = 24816  111110102 🡺 FA16  + 11108 🡺 + 24816  **34216** |

1. 306416 x 2138 (Show your final answer in Hex format.) (PYP-08/12: 3 marks)

|  |
| --- |
| **Answer** |
| 2138 = 010 001 0112  =0/1000/1011  = 0 8 B16    3 0 6 416  X 8 B16  2 1 4 4 C  1 8 3 2 0  **1 A 4 6 4 C16** |

**TUTORIAL 2: Numerical Data Representation**

**Section A: Signed Number**

1. Under what circumstances the Two’s Complement is used?

|  |
| --- |
| **Answer** |
| Two’s complement method can be used to: -   * Represent negative value * Perform subtraction |

1. Convert the 8-bit binary number 11010111 into decimal number if the binary number is a(n):
2. Unsigned number (PYP-04/14: 1 mark)

|  |
| --- |
| **Answer** |
| 110101112 = (1x27)+(1 x26)+(0 x25)+(1 x24)+(0 x23)+(1 x22)+(1 x21)+(1 x20)  = 128+64+0+16+0+4+2+1  = **21510** |

1. Signed number (PYP-04/14: 2 marks)

|  |
| --- |
| **Answer** |
| 110101112 = (1x27)+(1 x26)+(0 x25)+(1 x24)+(0 x23)+(1 x22)+(1 x21)+(1 x20)  = -128+64+0+16+0+4+2+1  = **-4110** |

1. Differentiate between carry flag and overflow flag. Complete the following table.

|  |  |  |
| --- | --- | --- |
| **Answer** | | |
| Flag | Carry | Overflow |
| Definition | Occurs when the result of an arithmetic operation exceeds the fixed number of bits allocated, without regard to sign | Occurs when the result of an arithmetic operation does not fit into the fixed numbers of bits available for the result |
| Detect in signed or unsigned number? | Unsigned numbers | Signed numbers |
| How to detect? | Extra “1” bit generated | 1. Both operands have the same sign 2. The result has opposite sign |
| Example | (-4) + (-2)  1100 No overflow  +1110 Carry  11010 The result is correct | (+4) + (+6)  0100 Overflow  +0110 No carry  1010 The result is incorrect |

1. Assuming an 8-bit system is used, show how the following operation is solved through Two’s Complement method.

- 12410 - 610

Verify and comment the answer. (PYP-04/14: 5 marks)

|  |
| --- |
| **Answer** |
| 12410 = 0111 11002  One’s complement = 1000 0011  + 1  Two’s complement = 1000 01002 (-12410)  610 = 0000 01102  One’s complement = 1111 1001  + 1  Two’s complement = 1111 10102 (-610)  -12410-610 = (-12410) + (-610)  1000 01002 -12410  + 1111 10102  + - 610  (1)0111 11102  - 13010  0111 11102 = +12610  This is invalid.  Because both binary (0111 11102 = +12610) and decimal (- 13010) computation return different results. |

1. Assuming that an 8-bit system is being applied, perform the binary subtraction operation for the following decimal numbers using Two’s Complement method.

65 - 54

Verify your answer by showing the answer in signed decimal value. (PYP-08/13: 5 marks)

|  |
| --- |
| **Answer** |
| 65 10 = 0100 00012  5410 = 0011 01102  One’s complement = 1100 1001  + 1  Two’s complement = 1100 10102 (-5410)  6510-5410 = (6510) + (-5410)  0100 00012 6510  + 1100 10102  + - 5410  (1)0000 10112  1110  0000 10112 = 8+2+1 = +1110 |

1. Assuming an 8-bit system is used (i.e. the system uses 8 bits to represent an integer). Given the following decimal numbers:

-12 + -8

1. Solve the above operation using two’s complement method. (PYP-08/15: 5 marks)

|  |
| --- |
| **Answer** |
| 1210 = 0000 11002  To prove 1110 110 🡪 -20  20 = 0001 0100  1’s complement = 1110 1011  + 1  2’s complement = 1110 1100  So 1110 1100 = -20  Or refer to (b)  One’s complement = 1111 0011  + 1  Two’s complement = 1111 01002 (-1210)  810 = 0000 10002  One’s complement = 1111 0111  + 1  Two’s complement = 1111 10002 (-810)  -1210-810 = (-1210) + (-810)  1111 01002 - 1210  + 1111 10002  + - 810  (1)1110 11002  - 2010 |

1. Verify your answer by showing the answer in signed decimal value. (PYP-08/13: 3 marks)

|  |
| --- |
| **Answer** |
| 1110 11002 = -128+64+32+8+4  = -2010 |

1. Justify the validity of the answer obtained. (PYP-08/13: 1 mark)

|  |
| --- |
| **Answer** |
| This is valid.  Because both binary and decimal computation return the same result. |

1. Does overflow occur? Justify your answer. (PYP-08/13: 1 mark)

|  |
| --- |
| **Answer** |
| Overflow does not occur.  Because the result having the same sign as both operands. |

1. Assuming an 8-bit system is involve.
2. Solve the following operation using Two’s Complement method: (PYP-08/14: 5 marks)

(-910) + (-810)

|  |
| --- |
| **Answer** |
| 910 = 0000 10012  One’s complement = 1111 0110  + 1  Two’s complement = 1111 01112 (-910)  810 = 0000 10002  One’s complement = 1111 0111  + 1  Two’s complement = 1111 10002 (-810)  1111 01112 - 910  + 1111 10002  + - 810  (1)1110 11112  - 1710 |

1. Verify your answer by showing the answer in signed decimal value. (PYP-08/14: 1 mark)

|  |
| --- |
| **Answer** |
| 1110 11112 = -128+64+32+8+4+2+1  = - 1710 |

1. Justify the validity of the answer obtained. (PYP-08/14: 4 marks)

|  |
| --- |
| **Answer** |
| This is valid.  Because both binary and decimal computation return the same result. |

1. Does overflow or/and carry occur?

|  |
| --- |
| **Answer** |
| Overflow = No  Carry = Yes |

**Section B: Floating Point Number**

1. Perform the following number conversions. Show your conversion steps clearly. If the operation is illogical, explain the reason.
2. 30.3010 to Binary (PYP-08/14: 2 marks)

|  |
| --- |
| **Answer** |
| 3010 = 111102  0.30x 2 = 0.60  0.60 x 2 = 1.20  0.20 x 2 = 0.40  0.4 x 2 = 0.8  0.8 x 2 = 1.6  0.6 x 2  30.3010 = **11110.010012** |

1. 123.1235 to Decimal (PYP-08/14: 2 marks)

|  |
| --- |
| **Answer** |
| 123.1235 = (1 x 52)+(2 x 51)+(3 x 50)+ (1 x 5-1)+(2 x 5-2)+(3 x 5-3)  = (25+10+3)+(0.2+0.08+0.024)  = **38.30410** |

1. 100100011111.112 to Octal (PYP-08/14: 2 marks)

|  |
| --- |
| **Answer** |
| 100100011111.112  = 100/100/011/111.110  =  **4 4 3 7 . 68** |

1. Perform the following operations. Show your working steps clearly. If the operation is illogical, explain the reason.
2. Convert 6A.9610 to hexadecimal number

|  |
| --- |
| **Answer** |
| Illogical.  Because base-10 does not include A. |

1. Convert 1807.6510 into a hexadecimal number (PYP-08/13: 2 marks)

|  |
| --- |
| **Answer** |
| 16) 1807 -F  16) 112 –0  7  0.65 x 16= 10.4  .4 x 16= 6.4    1807.6510 = **70F.A616** |

1. Convert 101011.01112 into a decimal number (PYP-04/13: 2 marks)

|  |
| --- |
| **Answer** |
| 101011.01112  = (1 x 25) + (0 x 24) + (1 x 23) + (0 x 22) + (1 x 21) + (1 x 20).(0 x 2-1) + (1 x 2-2) + (1 x 2-3) + (1 x 2-4)  = (32+0+8+0+2+1)+(0+0.25+0.125+0.0625)  = **43.437510** |

1. 111100110011.110000012 + 20.510. Show your answer in Hex format. (PYP-08/12: 3 marks)

|  |
| --- |
| **Answer** |
| 111100110011.110000012  = 1111/0011/0011.1100/0001  = F33.C116  20 = 101002  0.5 x 2 =1.0  20.510 = 10100.12  = 1/0100.1000  = 14.816  F 3 3 . C 1H  + 1 4 . 8 0 H  **F 4 8 . 4 1 H** |

1. Given that:

* An Excess-52 notation is applied.
* The implied decimal point is at the beginning of the mantissa.
* A “5” is used to represent a positive number and a “9” is used to represent a negative number.

1. Convert -357.24610 to the SEEMMMMM format. (PYP-04/14: 2 marks)

|  |
| --- |
| **Answer** |
| -357.24610 = - 0.35724(6) x 103  = **9 55 35725** (SEEMMMMM format) |

1. Convert 55220311 to scientific notation.

|  |
| --- |
| **Answer** |
| 55220311 = **+0.20311 x 100**(scientific notation) |

1. Convert 95575321 to scientific notation.

|  |
| --- |
| **Answer** |
| 95575321 = **- 0.75321 x 103** (scientific notation) |

1. Convert 30.81510 to the SEEMMMMM format.

|  |
| --- |
| **Answer** |
| 30.81510 = + 0.30815 x 102  = **5 54 30815** (SEEMMMMM format) |

1. The following decimal numbers are stored in excess-50 floating point format. A “1”is used to represent a negative sign, and a “5” for positive sign.
2. Add these two numbers. Present your result in standard decimal sign-and-magnitude notation.

55020311

15375321

(PYP-08/13: 5 marks)

|  |
| --- |
| **Answer** |
| 1. Adjust exponent: 5 53 00020311 2. Addition : 1 53 75321 + 3. SEEMMMMM : 1 53 75300689 4. Sign-Magnitude : **- 0.75301 x 103** |

1. Multiply these two numbers. Present your result in standard decimal sign-and-magnitude notation.

15176323

15485496 (PYP-08/13: 5 marks)

|  |
| --- |
| **Answer** |
| 1. Adjust exponent: 51 + 54 – 50 = 55 = 105 2. Multiply : - 0.76323 x - 0.85496 = 0.65253(11208) 3. SEEMMMMM : 5 55 65253 4. Sign-magnitude : **0.65253 x 105** |

1. Find the difference of these two numbers. Present your result in standard decimal sign-and-magnitude notation.

55152295

15256608 x – (-y) = x+y (PYP-04/14: 5 marks)

|  |
| --- |
| **Answer** |
| 1. Adjust exponent : 5 52 052295 2. Addition : 1 52 566080 - 3. SEEMMMMM : 5 52 61838 4. Sign-Magnitude : **+0.61838 x 102** |

1. The floating point decimal numbers below are stored in the form of SEEMMMMM where the exponent is stored in excess-50 with the implied decimal point at the beginning of the mantissa. A 4 in the sign position indicates a positive number and a 3 indicates a negative number:

45320460

35520112

1. Add these two numbers. Show the result in sign-magnitude notation. (PYP-08/12: 3 marks)

|  |
| --- |
| **Answer** |
| 1. Adjust exponent : 4 55 0020460 2. Addition : 3 55 20112 + 3. SEEMMMMM : 3 55 19907(40) 4. Sign-Magnitude : **-0.19907 x 105** |

1. Multiply these two numbers. Show the result in sign-magnitude notation. (PYP-08/12: 3 marks)

|  |
| --- |
| **Answer** |
| 1. Adjust exponent: 53 + 55 – 50 = 58 = 108 2. Multiply : 0.20460 x - 0.20112   = - 0.041149(152)  = - 0.41149 x 10-1   1. Sign-magnitude : - 0.41149 x 10-1 x108   = **- 0.41149 x 107**   1. SEEMMMMM : 3 57 41149 |

**Extra Example on Divide:**

05220000

04712500

1) adjust exponent : 52-47+50 = 55

2) divide : 0.22/.125

: 1.76

: 0.176 x 101

3) sign-magnitude : .176 x 101 x 105

: .176 x 106

4) SEEMMMMM : 05617600

1. Show how the number -5.510 is stored in the computer’s storage using IEEE754 32-bit single precision format. You are required to show your conversion steps clearly. (PYP-01/14: 6 marks)

|  |
| --- |
| **Answer** |
| -5.510 = -101.12   1. Adjust exponent : -1.011 x 22 2. Exponent (D) : 127 + 2 = 129 3. Exponent (B) : 1000 0001 4. Sign : 1 5. IEEE754 (S.P) : **1 1000 0001 0112..** |

1. Represent the binary number -10111.012 into IEEE754 single precision format. You are required to show your conversion steps clearly. (PYP-08/15: 5 marks)

|  |
| --- |
| **Answer** |
| -10111.012   1. Adjust exponent : -1.011101 x 24 2. Exponent (D) : 127 + 4 = 131 3. Exponent (B) : 1000 0011 4. Sign : 1 5. IEEE754 (S.P) : **1 1000 0011 0111012..** |

1. Given a decimal number “-30.8125”, how this notation can be represented in the IEEE754 single precision notation. You are required to show your working steps. (PYP-08/12: 5 marks)

|  |
| --- |
| **Answer** |
| -30.812510 = -11110.11012   1. Adjust exponent : -1.11101101 x 24 2. Exponent (D) : 127 + 4 = 131 3. Exponent (B) : 1000 0011 4. Sign : 1 5. IEEE754 (S.P) : **1 1000 0011 111011012..** |

1. Given an IEEE754 single precision notation below, show how this notation can be represented in a sign-magnitude notation. You are required to show your working steps.

1 1000 0010 0100 1000 0000 0000 0000 000

Assuming that excess-127 is applied. (PYP-08/11: 5 marks)

|  |
| --- |
| **Answer** |
| 1 1000 0010 0100 1000 0000 0000 0000 000   1. IEEE754 (S.P) : 1 1000 0010 0100 1000 0000 0000 0000 000 2. Sign : 1 = negative 3. Exponent (B) : 1000 0010 4. Exponent (D) : 130 = 23 5. Binary number : -1.01001 x 23   = -1010.012   1. Decimal number: -10.2510 2. Sign-magnitude: **- 0.1025 x 102** |

**TUTORIAL 3: Addressing Data in Memory and Segment**

1. Explain each of the following terms:
2. Segment

|  |
| --- |
| **Answer** |
| Segment   * Segments are defined as special areas of memory containing the code, data and stack information. * They are used to keep track of locations of individual program segments. * They are categorized into 3 key types: Code segment, data segment and stack segment. * Each segment can be up to 64KB in size. |

1. Offset

|  |
| --- |
| **Answer** |
| Offset   * It is defined as the distance in bytes from the segment address to another location within the segment. * It uses 16-bit addressing range from 0000H to FFFFH |

1. Register

|  |
| --- |
| **Answer** |
| Register   * It is defined as high speed storage location inside the CPU. * It is used to store information temporarily. |

1. Explain the purpose of the following items:
2. Code segment (PYP-01/16: 1 mark)

|  |
| --- |
| **Answer** |
| * Hold the machine instructions. |

1. Code segment (CS) register (PYP-01/16: 1 mark)

|  |
| --- |
| **Answer** |
| * Contains starting address of a program’s code segment. |

1. Instruction pointer (IP) register (PYP-01/16: 1 mark)

|  |
| --- |
| **Answer** |
| * Contains the offset address of the next instruction that is to execute. * Indicates the current instruction within the currently executing code segment. |

1. Accumulator (AX)

|  |
| --- |
| **Answer** |
| It is used for operations involving I/O and most arithmetic. |

1. Count register (CX)

|  |
| --- |
| **Answer** |
| It contains a value to control the number for times a loop is repeated or used for computation. |

1. Parity flag (PF)

|  |
| --- |
| **Answer** |
| It is used for error checking. |

1. (a) List and explain FOUR (4) main segment in a CPU. (PYP-04/14: 4 marks)

|  |
| --- |
| **Answer** |
| * Code segment * Data segment * Extra segment * Stack segment |

(b) Explain a similarity and THREE (3) differences between a register and a Random Access Memory (RAM). (PYP-08/13: 5 marks)

|  |  |  |
| --- | --- | --- |
| **Answer** | | |
|  | Register | Memory |
| Similarity | Provide storage | |
| Differences: (1) Data stored | Normal-byte sequence | Reversed-byte sequence |
| (2) Referred by | Name | Address |
| (3) Location | Inside CPU | Outside CPU |

(c) Differentiate between a segment:offset address and an absolute address.

|  |  |  |
| --- | --- | --- |
| **Answer** | | |
|  | Segment:Offset address | Absolute address |
| Characteristic | Logical address | Physical address |
| Size | 16-bit : 16-bit | 20-bit |
| Aim | To provide real address | To directly reference a specific location in memory |

1. Perform the following calculations and show your answers in hexadecimal format. Draw a diagram for each of them to indicate how the respective answer would store in the AX register. You are required to show your working steps clearly.
2. 738 + 258 (PYP-08/11: 4 marks)

|  |
| --- |
| **Answer** |
| 738 + 258 = 1208  = 5016  AH: 00H  AL: 50H  AX register |

1. 11112 x 1112 x 112  (PYP-08/11: 4 marks)

|  |
| --- |
| **Answer** |
| 1 1 1 1  X 1 1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 0 1 0 0 1  X 1 1  1 1 0 1 0 0 1  1 1 0 1 0 0 1  1 0 0 1 1 1 0 1 12 = 13BH  AH: 01H  AL: 3BH  AX register: |

1. Given the information in the table below.

|  |  |
| --- | --- |
| Code segment (CS) register: 02B3H  Data segment (DS) register: 26D2H  Stack segment (SS) register: 09AFH | Base Pointer (BP) register: 2062H  Stack Pointer (SP) register: 0094H  Instruction Pointer (IP) register: 0025H |

1. Calculate the absolute address for the next instruction to be executed by the CPU.

|  |
| --- |
| **Answer** |
| Absolute address = (Segment address x 10H) + Offset Address  = (02B3H x 10H) + 0025H  = **02B55H** |

1. Calculate the corresponding 20-bit absolute memory address using the SS:SP.

(PYP-08/13: 4 marks)

|  |
| --- |
| **Answer** |
| Absolute address = (Segment address x 10H) + Offset Address  = (09AFH x 10H) + 0094H  = **09B84H** |

1. Given a 16-bit CS register consists of the hexadecimal value 12AB and the 16-bit IP register consists of the hexadecimal value 0020. Find the absolute address of the instruction and show the answer in 20-bit binary format. (PYP-08/14: 4 marks)

|  |
| --- |
| **Answer** |
| Absolute address = (Segment address x 10H) + Offset Address  = (12ABH x 10H) + 0020H  = **12AD0H**  = 0001 0010 1010 1101 0000 |
|  |

1. Complete the following table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Answer** | | | |
| Register | Definition | A high speed storage located inside CPU | |
| Characteristics | Used to store information temporarily | |
| Types of register | General purpose | Address | Status |
| Aim | Used for arithmetic & data movement operation | Used for indexed addressing | Used to indicate the status of various activities |
| Size | 8-bit / 16-bit | 16-bit | 1-bit |
| Examples | 16-bit: AX,BX,CX,DX  8-bit:  AH, AL, BH, BL, CH, CL, DH, DL | Segment:  CS, DS, ES, SS  Pointer:  IP, BP, SP  Index:  SI, DI | OF  DF  IF  TF  SF  ZF  AF  PF  CF |

1. Determine which register is/are used for the following purposes:-
2. When the result of an arithmetic or logical operation generates a result of zero.

|  |
| --- |
| **Answer** |
| ZF |

1. Used for error checking when there is a possibility that data might be altered or corrupted.

|  |
| --- |
| **Answer** |
| PF |

1. When the result of an unsigned arithmetic operation is too large to fit into the destination.

|  |
| --- |
| **Answer** |
| CF |

(d) Used for arithmetic and data movement.

|  |
| --- |
| **Answer** |
| AX |

(e) Used for counting loops.

|  |
| --- |
| **Answer** |
| CX |

1. Write down the values of the Carry, Sign, Zero and Overflow flags after each instruction has executed. Show your working steps clearly.

MOV AX, 6120H

ADD AL, AAH

ADD AH, FFH

ADD AX, 2

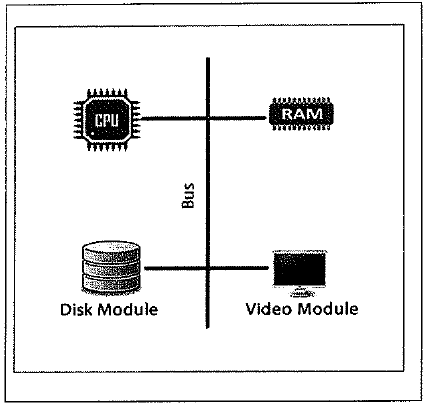
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Answer** | | | | | |
| Instructions | Operations | CF | SF | ZF | OF |
| MOV AX, 6120H | AX = 0110 0001 0010 00002 | NC | PL | NZ | NV |
| ADD AL, AAH | AL = 0010 00002  AAH = 1010 10102 +  AL = 1100 10102 | NC | NG | NZ | NV |
| ADD AH, FFH | AH = 0110 00012  FFH = 1111 11112 +  AH = (1)0110 00002 | CY | PL | NZ | NV |
| ADD AX, 2 | AX = 0110 0000 1100 10102  2 = 0000 0000 0000 00102 +  0110 0000 1100 11002 | NC | PL | NZ | NV |

1. Fill in the flag mnemonics in the following table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Answer** | | | |
| Set | Flag mnemonics | Clear | Flag mnemonics |
| Overflow | OV | No overflow | NV |
| Direction down | DW | Direction up | UP |
| Interrupt enabled | EI | Interrupt disabled | DI |
| Sign flag negative | NG | Sign flag positive | PL |
| Zero | ZR | Non zero | NZ |
| Auxiliary carry | AC | No auxiliary carry | NA |
| Odd parity | PO | Even parity | PE |
| carry | CY | No carry | NC |

**TUTORIAL 4: Computer Architecture**

1. Given the Figure 1 below.

Figure 1.

1. Define Bus in the computer context. (PYP-01/16: 3 marks)

|  |
| --- |
| **Answer** |
| Bus   * Bus is defined as the physical connection. * It is able to transfer data from one location to another in the computer system. * It aims to reduce the number of pathways needed for communication. |

1. Name the bus configuration as shown in the Figure 1 above. (PYP-01/16: 2 marks)

|  |
| --- |
| **Answer** |
| Multipoint bus. |

1. List and explain THREE (3) possible types of bus line that can be found in a multipoint bus configuration. (PYP-01/16: 6 marks)

|  |
| --- |
| **Answer** |
| Bus line   * Data bus: Carry the data that is being moved from one location to another. * Address bus: Specify the recipient of data on the bus. * Control bus: Provide control for the proper synchronization and operation of the bus and of modules that are connected to the bus. |

1. Briefly describe TWO (2) differences of point-to-point bus and multipoint bus. Draw a diagram to illustrate each of them. (PYP-04/10: 6 marks)

|  |  |  |
| --- | --- | --- |
| **Answer** | | |
|  | Point-to-point bus | Multipoint bus |
| Definition | A bus carry signals from a specific source to a specific destination | A bus that connect several points together |
| Bus lines | Data bus & control bus | Data bus, address bus & control bus |
| Diagram | Control unit 🡨🡪ALU | Computer Computer  Computer Computer |

1. *Memory registers play an important role in the process of Central Processing Unit (CPU) – Random Access Memory (RAM) communication.”*

Explain how memory Address Register (MAR) and Memory Data Register (MDR) involved in the CPU-RAM communication. (PYP-01/16: 5 marks)

|  |
| --- |
| **Answer** |
| * MAR is used to hold the memory address that is to be opened for memory data. * MDR is used to hold the data which is being retrieved / to be stored from / to memory location which is currently addressed by MAR. * Both MAR and MDR work together to provide interface for the CPU to communicate with the RAM. |

1. In the Little Man Computer (LMC) model, Instruction Register (IR), Program Counter (PC), Memory Address Register (MAR), Memory Data Register (MDR) and Accumulator (A) were used.
2. Describe the purpose for IR, PC and A respectively. (PYP-01/14: 6 marks)

|  |
| --- |
| **Answer** |
| * IR: Holds the instructions that are currently executed. * PC: Found in CU. Holds the address of the instruction that is going to be executed. * A: Found in ALU. Holds data / result that are used for arithmetic operations / transfer. |

1. How do IR, PC, A, MAR and MDR relate to each other in the operation of LMC?

(PYP-08/12: 3 marks)

|  |
| --- |
| **Answer** |
| * PC holds the MAR of the instruction that is going to be executed. * The MDR retrieved indicates the IR that is currently executed. * Based on the address IR, the address for operation is defined. * Load the MDR of the MAR to A. * The current operation is completed and the PC for next instruction is defined. |

1. Give a scenario or example to support your elaboration in Q4 (b) above.

(PYP-08/12: 4 marks)

|  |
| --- |
| **Answer** |
| PC 🡪 MAR  MDR 🡪 IR  IR [Address] 🡪 MAR  MDR 🡪 A  PC + 1 🡪 PC |

1. One large modern computer has a 36-bit MAR. How much memory can this computer address?

|  |
| --- |
| **Answer** |
| Using the formula: 2n where n refers to the size of MAR.  236 |

1. Assuming that LMC model is applied. Suppose that the following instructions are found at the given location in memory:

Program counter: 20

Value in memory location 20: 550 (LOAD)

Value in memory location 21: 151 (ADD)

Value in memory location 22: 350 (STORE)

:

Value in memory location 50: 422

Value in memory location 51: 008

Show the changes of contents in IR, PC, MAR, MDR and A during the execution of Instruction 20, 21 and 22.

|  |  |
| --- | --- |
| **Answer** | |
| PC 🡪 MAR  MDR 🡪 IR  IR [Address] 🡪 MAR  MDR 🡪 A  PC + 1 🡪 PC | MAR = 20  IR = 550  MAR = 50  A = 422  PC = 21 |
| PC 🡪 MAR  MDR 🡪 IR  IR [Address] 🡪 MAR  A + MDR 🡪 A  PC + 1 🡪 PC | MAR = 21  IR = 151  MAR = 51  A = 422 + 008 = 430  PC = 22 |
| PC 🡪 MAR  MDR 🡪 IR  IR [Address] 🡪 MAR  A 🡪 MDR  PC + 1 🡪 PC | MAR = 22  IR = 350  MAR = 50  MDR = 430  PC = 23 |

1. Show the changes of contents in IR, PC, MAR, MDR and A during the execution of Instruction 20 and Instruction 21. (PYP-08/15: 8 marks)

Program counter: 20

Value in memory location 20: 550 (LOAD)

Value in memory location 21: 151 (MUL)

Value in memory location 22: 350 (STORE)

:

Value in memory location 50: 2316

Value in memory location 51: 516

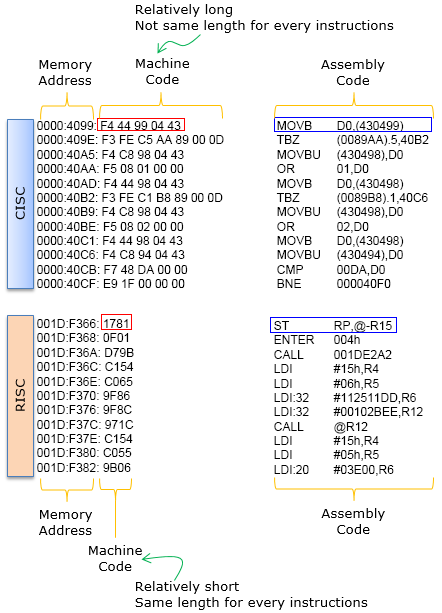
|  |  |
| --- | --- |
| **Answer** | |
| PC 🡪 MAR  MDR 🡪 IR  IR [Address] 🡪 MAR  MDR 🡪 A  PC + 1 🡪 PC | MAR = 20  IR = 550  MAR = 50  A = 2316  PC = 21 |
| PC 🡪 MAR  MDR 🡪 IR  IR [Address] 🡪 MAR  A x MDR 🡪 A  PC + 1 🡪 PC | MAR = 21  IR = 151  MAR = 51  A = 2316 x 516 = AF16  PC = 22 |
| If continue at instruction 22:  PC 🡪 MAR  MDR 🡪 IR  IR [Address] 🡪 MAR  A 🡪 MDR  PC + 1 🡪 PC | MAR = 22  IR = 350  MAR = 50  MDR = AF16  PC = 23 |

1. Most modern computers provide a large number of general-purpose registers and very few memory access instructions.
2. What is the computer architecture mentioned? (PYP-08/13: 2 marks)

|  |
| --- |
| **Answer** |
| Reduced instruction set computer (RISC ) |

1. Discuss any FOUR (4) advantages of such an architecture. (PYP-08/13: 12 marks)

|  |
| --- |
| **Answer** |
| 1. Limited and simple instruction set  * Execute faster at high clock speed * Do not require complex hardware  1. Registers-oriented instruction  * Reduce memory access * Use registers to operate / hold frequently used instruction  1. A fixed length / format instruction word  * Easy to identify * Can be fetched and decoded independently 🡪 pipelining  1. Limited addressing mode  * Provide single address mode * Speed up instruction executions  1. A large bank of registers  * Registers are applied widely * Reduce memory access |



**TUTORIAL 5: Machine Execution**

* + - 1. Name the FOUR (4) basic components of an assembly language instruction.

|  |
| --- |
| **Answer** |
| [label: ] operation operand(s) [;comment] |

* + - 1. What is the rule to follow when the following types of operand is used?

1. Memory as operand

|  |
| --- |
| **Answer** |
| * For 2-operand instructions, memory to memory is not allowed * Example:   ADD AH, NUM 🡪 allow  ADD NUM1, NUM2 🡪 not allow |

1. Register as operand

|  |
| --- |
| **Answer** |
| * Both registers must be the same size * Example:   ADD AH, BL |

1. Immediate value as operand

|  |
| --- |
| **Answer** |
| * For 2-operand instructions, constants can appear only as the second operand * Example:   ADD AH, 40H |

* + - 1. Issue DEBUG commands for the following operations:

1. Reset current IP value to 200. (PYP-08/14: 2 marks)

|  |
| --- |
| **Answer** |
| * R IP   IP XXXX  :200 |

1. Insert the machine instruction 10 AB 2A 35 and 5D into Code Segment with Offset Address 100. (PYP-08/14: 2 marks)

|  |
| --- |
| **Answer** |
| E CS:100 10 AB 2A 35 5D |

1. Translate the machine instruction from Offset Address 100 to 108 into assembly code.

(PYP-08/14: 2 marks)

|  |
| --- |
| **Answer** |
| U 100, 108 |

1. Execute a group of 5 instruction starts from the offset address 100H. (PYP-08/11: 2 marks)

|  |
| --- |
| **Answer** |
| P=100,5 |

1. Compare the content at code segment in offset address range 0100H to 010AH with the content in the offset address range 0150H to 015AH (PYP-08/11: 2 marks)

|  |
| --- |
| **Answer** |
| C CS:100 10A 150 |

* + - 1. Explain the following DEBUG operations.

1. –D CS:200 (PYP-01/14: 1 mark)

|  |
| --- |
| **Answer** |
| Display the content of memory at code segment starts from offset address 0200H |

1. –A 200

XXXX:0200 SUB BL,42 (PYP-01/14: 2 marks)

|  |
| --- |
| **Answer** |
| Line 1: Assemble (convert symbolic code to object code) at the code segment starts from offset address 0200H.  Line 2: At the memory address CS:0200, perform subtraction. BL = BL – 42. |

1. –T 3 (PYP-04/14: 1 mark)

|  |
| --- |
| **Answer** |
| Execute a group of 3 instructions at once. |

1. H 3064 2130

|  |
| --- |
| **Answer** |
| Perform Hex computation in addition and subtraction.  Result: 5194 (Result of addition) 0F34 (Result of subtraction) |

* + - 1. Provide the symbolic code for the following operations and trace the execution result for each of them.

1. Move the hexadecimal value 4269 to AX register

|  |  |
| --- | --- |
| **Answer** | |
| MOV AX,4269 | AX = 4269H |

1. Move the 7012H to BX register

|  |  |
| --- | --- |
| **Answer** | |
| MOV BX,7012 | BX = 7012H |

1. Add the value of BX to AX register

|  |  |
| --- | --- |
| **Answer** | |
| ADD AX,BX | AX = AX + BX  = 4269H + 7012H  = B27BH |

1. Exchange the content in AX and BX

|  |  |
| --- | --- |
| **Answer** | |
| XCHG AX,BX | AX BX  Before exchange: B27BH 7012H  After exchange : 7012H B27BH |

* + - 1. Describe each of the following DEBUG commands. Complete the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Answer** | | | |
| Command | Purpose | Format / E.g. | Output |
| A | Convert symbolic code to object code | -A 100  XXXX:0100 | Object codes. (Not shown.) |
| C | Compare 2 groups of content in memory segment | -C CS:100 105 120 | Segment:offset addresses & content of the selected groups |
| D | Display the content of memory segment | -D CS:100 | Segment:offset addresses, object codes & ASCII codes |
| E | Enter the object to memory segment | -E CS:100 8B | Not shown.  Use D command to confirm entered |
| F | Fill in a group of ASCII code repeatedly | -F CS:100 13F “CSA” | Not shown.  Use D command to confirm filled in. |
| H | Perform hexadecimal computation | -H 5555 2222 | Result of addition and subtraction |
| P | Proceed to execute a group of instruction | -P=100, 5 | Changes of registers & next instruction |
| Q | Quit the DEBUG program | -Q | Back to root directory |
| R | Retrieve / edit the content of register(s) | -R  -R AX  AX 0000  :200 | Content of register(s) |
| T | Trace the execution of instruction 1 by 1 | -T  -T3 | Changes of registers & next instruction |
| U | Convert object code to symbolic code | U 100, 108 | Segment:offset address, symbolic codes & object codes |

* + - 1. Assume that you have used DEBUG program to enter the following E command :

E CS:100 B8 45 01 05 25 00

1. Write the command to change the hex value 45 to 54.

|  |
| --- |
| **Answer** |
| -E CS:101 54 |

1. Write the command to change the instruction pointer address to 200H.

|  |
| --- |
| **Answer** |
| -R IP  IP XXXX  :200 |

* + - 1. Assume that the IP register has value 0100H. What is its value after the instruction B8 25 00 is executed?

|  |
| --- |
| **Answer** |
| IP = 0100H  Instruction = B82500H (3 bytes)  New IP = 0100h + 3  = 0103H |

**TUTORIAL 6: Computer Architecture**

1. Write the command line to perform the followings:
2. To assemble a source program named CSA.ASM into listing and object files.

|  |
| --- |
| **Answer** |
| MASM CSA.ASM |

1. To link an object file named CSA.OBJ into executable and map files.

|  |
| --- |
| **Answer** |
| LINK CSA.OBJ |

1. To directly execute CSA.EXE from DOS.

|  |
| --- |
| **Answer** |
| CSA  ---Or---  CSA.EXE |

1. To execute CSA.EXE through DEBUG program.

|  |
| --- |
| **Answer** |
| TD CSA.EXE |

1. Briefly describe the function of TWO-PASS ASSEMBLER in the process of assembling an Assembly Language program. (PYP-08/12: 3 marks)

|  |
| --- |
| **Answer** |
| Two-pass assembler   * Aim: To resolve forward reference to addresses not yet encountered in the program. * Process: * Pass 1: The assembler reads the entire source program and construct a symbol table of names and labels used in the program. The amount of codes to be generated for each instruction is determined during Pass 1. * Pass 2: The assembler uses the symbol table that is constructed in Pass 1 and complete with the object code for each instruction. It then produces, on request, the object (.OBJ), list (LST) and cross reference (.CRF) files. |

1. Illustrate the process of editing, assembling, linking and executing assembly language programs in an appropriate diagram

|  |
| --- |
| **Answer** |
| Step 1: Edit (Editor)  .ASM  Step 2: Assemble (Assembler)  .OBJ  .LST  .CRF  Step 3: Link (Linker)  .EXE  Step 4: Load (Loader)  Memory |

1. Write an assembly program using simplified segment definitions to perform the following operations:
2. Move the immediate value hex 40 to the AL register
3. Shift AL contents one bit left
4. Move immediate value hex 1A to BL
5. Multiply AL by BL

|  |  |
| --- | --- |
| **Answer** | |
| .MODEL SMALL  .STACK 64  .CODE  MAIN PROC  MOV AL, 40H  SHL AL, 1  MOV BL, 1AH  MUL BL  MOV AX,4C00H  INT 21H  MAIN ENDP  END MAIN | ; (a) AL = 40H  ; (b) AL = 40H = 0100 0000B 🡺 1000 0000B = 80H  ; (c) BL = 1AH  ; (d) AX = AL \* BL 🡺 80H \* 1AH = 0D00H |

1. Modify your answer in Q4 above for the following requirements:
2. Define a 1-byte item named ITEMA containing hex 40 and another named ITEMB containing hex 1A.
3. Define a 2-byte item named ITEMC with no constant
4. Move the contents of ITEMA to AL and shift left one bit
5. Multiply AL by ITEMB
6. Move the product in AX to ITEMC

What is the content in the destination register for each of the above instruction? Show your working.

|  |  |
| --- | --- |
| **Answer** | |
| .MODEL SMALL  .STACK 64  .DATA  ITEMA DB 40H  ITEMB DB 1AH  ITEMC DW 0  .CODE  MAIN PROC  MOV AX,@DATA  MOV DS,AX  MOV AL, ITEMA  SHL AL, 1  *MUL ITEM B*  MOV ITEMC, AX  MOV AX,4C00H  INT 21H  MAIN ENDP  END MAIN | ; (a)  ; (a)  ; (b)  ; (c) AL = ITEM1 = 40H  ; (c) AL = 40H = 0100 0000B 🡺 1000 0000B = 80H  ; (d) AX = AL \* ITEMB 🡺 80H \* 1AH = 0D00H  ; (e) ITEMC = AX = 0D00H |

**TUTORIAL 7: Assembly Language Fundamentals 1**

1. Define the following data items in assembly language with the name DATA1 to DATA4 respectively.

(PYP-01/16: 4 marks)

1. A 1-byte item containing the hex equivalent to decimal 40.

|  |
| --- |
| **Answer** |
| DATA1 DB 28H |

1. A 2-byte item containing an undefined value.

|  |
| --- |
| **Answer** |
| DATA2 DW 0 |

1. A string with the characters H,E,L,L,O

|  |
| --- |
| **Answer** |
| DATA3 DB "H","E","L","L","O",”$”  DATA3 DB “HELLO$” |

1. An item containing the consecutive word values 1, 3, 5, 7 and 9.

|  |
| --- |
| **Answer** |
| DATA4 DW 1,3,5,7,9 |

1. Determine whether each of the following names is VALID or INVALID identifier that conforms to the rules of Assembly Language. Give the reason if it is INVALID. (PYP-01/09: 10 marks)
2. 2NAME
3. Total\_1
4. #TelNo
5. .Counter
6. @Word

|  |  |  |  |
| --- | --- | --- | --- |
| **Answer** | | | |
| (a) | 2NAME | Invalid | First character cannot be a digit |
| (b) | Total\_1 | Valid | - |
| (c) | #TelNo | Valid | - |
| (d) | .Counter | Invalid | First character cannot be a dot “.” |
| (e) | @Word | Invalid | avoid using “@” as the first character because the assembler uses some special words that begin with “@” |

1. Below shows a sample of an assembly program. Examine the code. Identify SIX (6) errors found in the program and correct them. You are required to indicate the Line Number in your answer followed by the correct code statements. (PYP-04/13: 6 marks)

|  |  |
| --- | --- |
| Line number | Program content |
| Line 1  Line 2  Line 3  Line 4  Line 5  Line 6  Line 7  Line 8  Line 9  Line 10  Line 11  Line 12  Line 13  Line 14  Line 15  Line 16  Line 17 | . MODEL SMALL  . STACK 64  . DATA  DATA1 DW 1AH  DATA2 DB 380  DATA3 DW ?  .CODE  MAIN PROC FAR  MOV AX, DATA  MOV DS, AX  MOV AX, DATA1  ADD AX, DATA2  MOV DATA3, AX  MOV FX, 4C00H  INT 21H  MAIN END  ENDP MAIN |

|  |  |  |
| --- | --- | --- |
| **Answer** | | |
| Line Number | Error | Correction |
| 5 | DB | DW (DB🡪max 255) |
| 9 | DATA | @DATA |
| 12 | AX | AL 🡪no error, if line 5changed to DW |
| 14 | FX | AX |
| 16 | END | ENDP |
| 17 | ENDP | END |

1. Determine the final content of AL register after running the following assembly instructions.

MOV AX, 0H

MOV AL, 34H

MOV BL, 04H

MOV CL, AL

DIV CL

MUL BL

SHL AL, 1

DEC AL (PYP-08/15: 2 marks)

|  |  |
| --- | --- |
| **Answer** | |
| MOV AX, 0  MOV AL, 34  MOV BL, 04  MOV CL, AL  DIV CL  MUL BL  SHL AL, 1  DEC AL | ; AX = 0000H  ; AL = 34H  ; BL = 04H  ; CL = 34H  ; AX = AX / CL 🡺 AX = 34/34 = 0001H  ; AX = AX \* BL 🡺 AX = 0001 \*04H = 0004H  ; AL = 04H 🡺 0000 0100🡪 0000 1000B 🡪 08H  ; AL = AL – 1 🡺 07H |

1. Consider the code fragment below. Examine the code and show the contents of the AX and BX registers. (PYP-08/13: 10 marks)

|  |
| --- |
| .DATA  DATA1 DB 32H  DATA2 DB 12H  .CODE  MOV AX, 0004H  MOV BX, 0000H  MOV BL, DATA1 ; AX= ? , BX = ?  SHR BL, 1 ; AX= ? , BX = ?  ADD BL, DATA2 ; AX= ? , BX = ?  MUL AL ; AX= ? , BX = ?  SUB BX, AX ; AX= ? , BX = ? |

|  |  |
| --- | --- |
| **Answer** | |
| .DATA  DATA1 DB 32H  DATA2 DB 12H  .CODE  MOV AX, 0004H  MOV BX, 0000H  MOV BL, DATA1  SHR BL, 1  ADD BL, DATA2  MUL AL  SUB BX, AX | AX = 0004H, BX = 0000H  AX = 0004H, BX = 0032H  AX = 0004H, BX = 0019H  AX = 0004H, BX = 002BH  AX = 0010H, BX = 002BH  AX = 0010H, BX = 001BH |

1. Using only MOV, ADD, SUB, INC, DEC, and NEG, translates the following high-level language assignment statements into assembly language. Assume that A, B, and C are word variables.
2. A = B – A

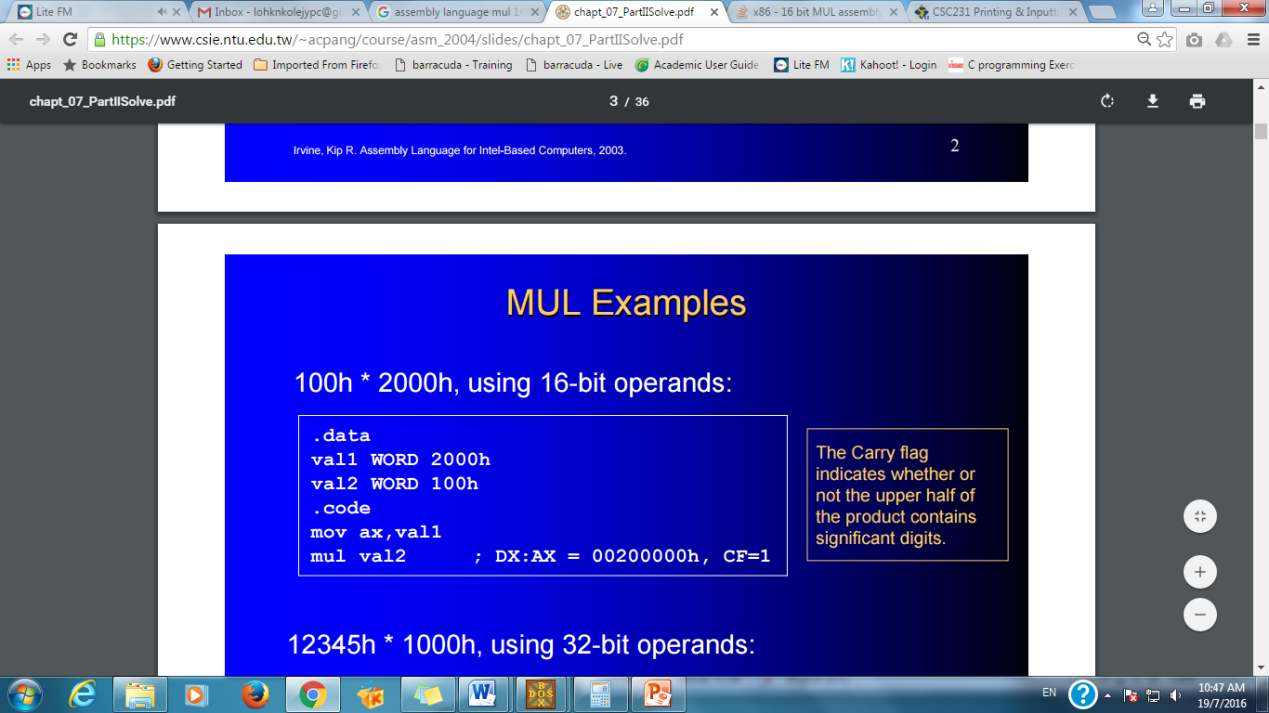
|  |
| --- |
| **Answer** |
| MOV BX, B  SUB BX, A  MOV A,BX |

1. C = A + B

|  |
| --- |
| **Answer** |
| MOV AX, A  ADD AX, B  MOV C, AX |

1. A = - (A + 1)

|  |  |
| --- | --- |
| **Answer 1** | Answer 2 |
| MOV AX, A  ADD AX, 1 or INC AX  NEG AX  MOV A , AX | INC A  NEG A |

1. B = 4 \* B + 6

|  |  |  |
| --- | --- | --- |
| **Answer 1** | **Answer 2** | **Answer 3** |
| MOV AX, B  MOV CX, 3  CALC:  ADD AX, B  LOOP CALC  ADD AX, 6  MOV B, AX | MOV AX, 4  MUL B  ADD AX, 6  MOV B, AX | MOV BX, B  ADD BX,B  ADD BX,B  ADD BX,B  ADD BX,6  MOV B,BX |

1. A = B - A – 1

|  |
| --- |
| **Answer** |
| MOV BX, B  SUB BX, A  DEC BX or SUB BX, 1  MOV A, BX |

1. Construct an Assembly Language program using simplified segment directive to calculate the average value based on your inputs. Your program should fulfill the following requirements.

(PYP-08/12: 10 marks)

Marks will be awarded according to the following criteria.

|  |  |
| --- | --- |
| Requirements | Mark allocation |
| Prompt and accept user input of TWO (2) odd integers. Display a comma (,) automatically after the user keys in the first input where the comma is used to serve as a separator between the first and second input | 3 marks |
| Calculate the average using the formula below: (Input1 + Input2) / 2 | 3 marks |
| Calculate and display the average in 1 decimal place | 2 marks |
| Overall program structure | 2 marks |
| Total: | 10 marks |

|  |
| --- |
| Sample output : |
| Enter TWO odd integer values (0 to 9): 5,7  The average of the input values is: 6.0 |

|  |  |
| --- | --- |
| .MODEL SMALL  .STACK 100  .DATA  MSG1 DB "Enter TWO odd integer values (0 to 9): $"  MSG2 DB 0DH,0AH, "The average of the input values is: $"  INPUT1 DB 0  INPUT2 DB 0  Q DB ?  R DB ?  TWO DB 2  .CODE  MAIN PROC  MOV AX,@DATA  MOV DS,AX  ;---------Display prompt  MOV AH,09H  LEA DX,MSG1  INT 21H  ;---------Get input  MOV AH,01H  INT 21H  SUB AL,30H  MOV INPUT1,AL  MOV AH,02H  MOV DL,","  INT 21H  MOV AH,01H  INT 21H  SUB AL,30H  MOV INPUT2,AL | ;--------Output message  MOV AH,09H  LEA DX,MSG2  INT 21H  ;---------Calculate average  MOV AL,INPUT1  ADD AL,INPUT2  CBW  DIV TWO  MOV Q,AL  MOV R,AH    ;-------Output result  MOV AH,02H  MOV DL,Q  ADD DL,30H  INT 21H  MOV AH,02H  MOV DL,"."  INT 21H  MOV AH,02H  MOV DL,R  ADD DL,30H  INT 21H  MOV AX,4C00H  INT 21H  MAIN ENDP  END MAIN |

**TUTORIAL 8: Assembly Language Fundamentals II**

1. Differentiate among the following addresses by completing the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Answer** | | | |
|  | Short | Near | Far |
| Distance covered | -128 to 127  (Same segment) | -32768 to 32767  (Same segment) | Another segment |
| Instruction(s) used | JMP  Jnnn🡪 JE,JNLE  LOOP | JMP  Jnnn (for 80386/486) | JMP |

1. Using *int 21h* function *01h* and *LOOP* instruction, write a segment of assembly program to accept 3 characters and store them into a variable named INPUT. (PYP-08/15: 3 marks)

.DATA

INPUT DB 3 DUP (" "), "$"

.CODE

; - Write the assembly codes here –

|  |
| --- |
| **Answer** |
| .DATA  INPUT DB 3 DUP (0)  .CODE  MOV CX, 3  MOV SI,0  getInput:  MOV AH,01H  INT 21H  MOV INPUT[SI], AL  INC SI  LOOP getInput |

1. Based on the assembly program shown below, answer the following questions:

|  |  |
| --- | --- |
| Line number | Program Contents |
| Line 1  Line 2  Line 3  Line 4  Line 5  Line 6  Line 7  Line 8  Line 9  Line 10  Line 11  Line 12  Line 13  Line 14  Line 15  Line 16  Line 17  Line 18 | . MODEL SMALL  .STACK 64  .DATA  MESSAGE1 DW "HI ASSEMBLY!", 10, 13, "$"  -----------------------------------------------------------------------  .CODE  MAIN PROC FAR  MOV AX, DATA  MOV DX, AX  MOV CX, 9  L1: MOV AH, 09H  LEA DS, MESSAGE1  INT 21H  LOOP L1  MOV AX, 4C00H  INT 21H  MAIN END  END MAIN |

1. The assembly program consists of FIVE (5) errors. Identify and correct the errors.

(PYP-01/14: 5 marks)

|  |  |  |
| --- | --- | --- |
| **Answer** | | |
| Line No | Error | Correction |
| 5 | ----- | ;----- |
| 8 | DATA | @DATA |
| 9 | DX | DS |
| 12 | DS | DX |
| 17 | END | ENDP |

1. How many times will the message “HI ASSEMBLY!” be displayed on the screen when the program runs? (PYP-01/14: 1 mark)

|  |
| --- |
| **Answer** |
| 9 times |

1. Given the following variable definition in the data segment.

|  |  |
| --- | --- |
|  |  |
| Data definition | Sample output |
| .DATA  VAR1 DB 2, 0, 1, 1 | 2011 |

Write a program segment to display consecutively each byte of the data defined in VAR1 using indirect-offset address with LOOP operator. (PYP-08/11: 4 marks)

|  |
| --- |
| **Answer** |
| .DATA  VAR1 DB 2, 0, 1, 1  .CODE  MOV CX, 4  MOV SI,0  printOutput:  MOV AH, 02H  MOV DL, VAR1[SI]  ADD DL,30H  INT 21H  INC SI  LOOP printOutput |
|  |

1. Identify and explain how many times each of the following LOOP operations loop:
2. MOV CX , 1

L1 :

….

LOOP L1

|  |  |
| --- | --- |
| **Answer** | |
| 1 time | Because CX = 1 |

1. L2 : MOV CX , 10

…..

LOOP L2

|  |  |
| --- | --- |
| **Answer** | |
| Infinitive | Because CX is reset within the loop |

1. MOV CX , 10

L3 :

INC CX

LOOP L3

|  |  |
| --- | --- |
| **Answer** | |
| Infinitive | Because CX will never reach zero |

1. Determine the value in the destination register and the result in the Flags register for OF, ZF, SF, and CF.
2. MOV AL , FFH

ADD AL , 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Answer** | | | | | |
| Instruction | Operation | OF | ZF | SF | CF |
| MOV AL , FFH | AL = FFH  = 1111 1111B | NV | NZ | NG | NC |
| ADD AL , 1 | AL = 1111 1111B  ADD = 0000 0001B +  AL =(1)0000 0000B  = 00H | NV | ZR | PL | CY |

1. MOV BL , 24H

SUB BL , BL

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Answer** | | | | | |
| Instruction | Operation | OF | ZF | SF | CF |
| MOV BL , 24H | BL = 24H  = 0010 0100B | NV | NZ | PL | NC |
| SUB BL , BL | BL = 0010 0100B  SUB = 0010 0100B -  BL = 0000 0000 | NV | ZR | PL | NC |

1. MOV CL , 10101010B

ADD CL , 01010101B

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Answer** | | | | | |
| Instruction | Operation | OF | ZF | SF | CF |
| MOV CL , 10101010B | CL = 10101010B | NV | NZ | NG | NC |
| ADD CL , 01010101B | CL = 1010 1010B  ADD = 0101 0101B +  CL = 1111 1111B | NV | NZ | NG | NC |

1. MOV DL, 11001100B

ADD DL, 01110011B

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Answer** | | | | | |
| Instruction | Operation | OF | ZF | SF | CF |
| MOV DL, 11001100B | DL = 11001100B | NV | NZ | NG | NC |
| ADD DL, 01110011B | DL = 1100 1100B  ADD = 0111 0011B +  DL =(1)0011 1111B | NV | NZ | PL | CY |

1. CHAR\_STRING contains uppercase letters that a program is to convert a lowercase. Access each character successively into a register, add 20H to it, and restore it in the string. Use indirect addressing and the LOOP instruction.

(Hint: Use indirect addressing and increment BX for each character)

CHAR\_STRING DB "ABCDEFGHIJ"

|  |
| --- |
| **Answer** |
| LEA BX , CHAR\_STRING  MOV CX, 10  L10:  MOV AL , [BX]  ADD AL , 20H  MOV [BX] , AL  INC BX  LOOP L10 |

8. Given the following table and initializing instructions, show the effect of the MOV

VALUE\_TBL DB 1, 2, 3, 4, 5, 6, 7, 8

…

LEA BX, VALUE\_TBL

MOV DI, 4

1. MOV CL, [BX] ; CL

|  |
| --- |
| **Answer** |
| 1 |

1. MOV DL, [BX + 3] ; DL

|  |
| --- |
| **Answer** |
| 4 |

1. MOV AL, [BX + DI] ; AL

|  |
| --- |
| **Answer** |
| 5 |

9. Using nested loop structure, write an assembly program that print out the right-angled triangle as below:

|  |
| --- |
| Sample output : |
| \*\*\*  \*\*  \* |

|  |  |
| --- | --- |
| **Answer** | |
| .MODEL SMALL  .STACK 100  .DATA    .CODE  MAIN PROC  MOV AX,@DATA  MOV DS,AX    MOV CX,3  MOV SI,1    ROW:  MOV AH,02H  MOV DL,0AH  INT 21H  MOV DL,0DH  INT 21H | MOV DI,CX  MOV CX,DI    COL:  MOV AH,02H  MOV DL,"\*"  INT 21H  LOOP COL    INC SI  MOV CX,DI  LOOP ROW    EXIT:  MOV AX,4C00H  INT 21H  MAIN ENDP  END MAIN |

10. Modify the assembly program in Q9 to generate the result as below:

|  |
| --- |
| Sample output : |
| 123  45  6 |

|  |  |
| --- | --- |
| **Answer** | |
| .MODEL SMALL  .STACK 100  .DATA  A DB 31H  .CODE  MAIN PROC  MOV AX,@DATA  MOV DS,AX    MOV CX,3  MOV SI,1    ROW: MOV AH,02H  MOV DL,0AH  INT 21H  MOV DL,0DH  INT 21H | MOV DI,CX  MOV CX,DI    COL: MOV AH,02H  MOV DL,A  INT 21H  INC A  LOOP COL    INC SI  MOV CX,DI  LOOP ROW    EXIT:MOV AX,4C00H  INT 21H  MAIN ENDP  END MAIN |

11. Modify the assembly program in Q10 to generate the result as below:

|  |
| --- |
| Sample output : |
| 12345  1234  123  12  1 |

|  |  |
| --- | --- |
| **Answer** | |
| .MODEL SMALL  .STACK 100  .DATA  A DB 31H  .CODE  MAIN PROC  MOV AX,@DATA  MOV DS,AX    MOV CX,6  MOV SI,1    ROW: MOV AH,02H  MOV DL,0AH  INT 21H  MOV DL,0DH  INT 21H | MOV DI,CX  MOV CX,DI    COL: MOV AH,02H  MOV DL,A  INT 21H  INC A  LOOP COL    INC SI  MOV CX,DI  MOV A,31H  LOOP ROW    EXIT:MOV AX,4C00H  INT 21H  MAIN ENDP  END MAIN |

**TUTORIAL 9: Conditional Processing**

1. Assume that AL and BL contain unsigned data and that CL and DL contain signed data. Determine the CMP and conditional jump instructions for the following.
2. Is CL equal to or smaller than DL?

|  |
| --- |
| **Answer** |
| CMP CL, DL  JLE L1 |

1. Is AL equal to or smaller than BL?

|  |
| --- |
| **Answer** |
| CMP AL, BL  JBE L1 |

1. Is AL greater than BL?

|  |
| --- |
| **Answer** |
| CMP AL, BL  JA L1 |

1. Is CL greater than DL?

|  |
| --- |
| **Answer** |
| CMP CL, DL  JG L1 |

1. Does DL contain zero?

|  |
| --- |
| **Answer** |
| CMP DL, 0  JZ L1 |

1. Is there an overflow?

|  |
| --- |
| **Answer** |
| CMP CL, DL  JO L1 |

1. Assume that DL contains 01111001 and that an item named BOOL\_AMT contains 11100011. Determine the effect on DL for the following unrelated operations:
2. AND DL , BOOL\_AMT

|  |
| --- |
| **Answer** |
| DL = 0111 1001B  BOOL\_AMT = 1110 0011B AND  DL = 0110 0001B |

1. OR DL , BOOL\_AMT

|  |
| --- |
| **Answer** |
| DL = 0111 1001B  BOOL\_AMT = 1110 0011B OR  DL = 1111 1011B |

1. XOR DL , BOOL\_AMT

|  |
| --- |
| **Answer** |
| DL = 0111 1001B  BOOL\_AMT = 1110 0011B XOR  DL = 1001 1010B |

1. AND DL , 00000000B

|  |
| --- |
| **Answer** |
| DL = 0111 1001B  0000 0000B AND  DL = 0000 0000B |

1. XOR DL , 11111111B

|  |
| --- |
| **Answer** |
| DL = 0111 1001B  1111 1111B XOR  DL = 1000 0110B |

1. Write instructions that first clear bit 0 and bit 1 in AL. Then, if the destination operand is equal to zero, the code should jump to label L3. Otherwise, it should jump to label L4.

|  |  |
| --- | --- |
| **Answer** | **LKN answer** |
| ; AL = 1010 1011B  SHR AL, 1 ; AL = 0101 0101B  SHR AL, 1 ; AL = 0010 1010B  SHL AL, 1 ; AL = 0101 0100B  SHL AL, 1 ; AL = 1010 1000B  JZ L3  JNZ L4 | AND AL, 1111 1100B  JZ L3  JNZ L4  OR SHR AL,2 SHL AL,2 |

1. Translate the following C statements into assembly language.
2. if (num1 == num2)

{

X = 1; Y = 2;

}

|  |
| --- |
| **Answer** |
| MOV BH, num1  CMP BH, num2  JNE L1  MOV X, 1  MOV Y, 2  L1: |

1. if (al > bl && bl > cl)

{

X = 1;

}

**Y = 2;**

**Z = 4;**

|  |  |  |
| --- | --- | --- |
| **Answer** | **LKN answer 1** | **LKN answer 2** |
| CMP AL, BL  JBE L1  CMP BL, CL  JA L2  L1:  …  L2:  MOV X, 1 | CMP AL, BL  JLE L1  CMP BL, CL  JLE L1  MOV X, 1  JMP FINISH  L1:  MOV Y,2  MOV Z,4 | CMP AL,BL  JG L1  JMP L3  L1: CMP BL,CL  JG L2  JMP L3  L2: MOV X,1  L3: MOV Y,2  MOV Z,4 |

1. if (al > bl || bl > cl)

{

X = 1;

}

|  |
| --- |
| **Answer** |
| CMP AL,BL  JA L1  CMP BL, CL  JA L1  JMP FINISH  L1:  MOV X, 1 |

1. Write an assembly program to count the vowels in a character array “This is my favorite”. Print out the number of occurrences of each of the vowels a, e, i, o, and u.

Sample output:

a =1 ;e = 1 ; i = 3 ; o = 1 ; u = 0

|  |  |
| --- | --- |
| **Answer** | |
| .MODEL SMALL  .STACK 100  .DATA  LIST DB "THIS IS MY FAVORITE$"  CA DB 30H  CE DB 30H  CI DB 30H  CO DB 30H  CU DB 30H  .CODE  MAIN PROC  MOV AX,@DATA  MOV DS,AX  MOV SI,0  **AGAIN:**  MOV BH, LIST[SI]    CMP BH, "a"  JE COUNTA    CMP BH, "e"  JE COUNTE    CMP BH, "i"  JE COUNTI    CMP BH, "o"  JE COUNTO    CMP BH, "u"  JE COUNTU    CMP BH,"$"  JE DISPLAY  INC SI  JMP AGAIN    COUNTA:  INC CA  INC SI  JMP AGAIN  COUNTE:  INC CE  INC SI  JMP AGAIN  COUNTI:  INC CI  INC SI  JMP AGAIN | COUNTO:  INC CO  INC SI  JMP AGAIN  COUNTU:  INC CU  INC SI  JMP AGAIN  ;--------------DISPLAY  **DISPLAY:**  MOV AH,02H  MOV DL,0DH  INT 21H  MOV DL,0AH  INT 21H    MOV AH,02H  MOV DL,CA  INT 21H  MOV AH,02H  MOV DL,CE  INT 21H  MOV AH,02H  MOV DL,CI  INT 21H  MOV AH,02H  MOV DL,CO  INT 21H  MOV AH,02H  MOV DL,CU  INT 21H    MOV AX,4C00H  INT 21H    MAIN ENDP  END MAIN |
|  |  |

1. Using simplified segment directive, write an assembly language program that will find the largest value from a list. (PYP-08/13: 15 marks)

The program should:

1. Prompt the user to enter 5 decimal digits.
2. Accept the user input. (Assume user will only enter 5 digits)
3. Find the largest value inside the list.
4. Display the largest value on the screen.

|  |
| --- |
| Sample output: |
| Please enter 5 decimal digits >> 35827  The largest value in the list is >> 8 |

|  |  |
| --- | --- |
| **Answer** | |
| .MODEL SMALL  .STACK 100  .DATA  INPUT DB 5 DUP (0)  LARGEST DB 0  .CODE  MAIN PROC  MOV AX,@DATA  MOV DS,AX    MOV CX,5  MOV SI,0    GETINPUT:  MOV AH,01H  INT 21H  MOV INPUT[SI],AL  INC SI  LOOP GETINPUT    MOV CL,INPUT[0]  MOV LARGEST,CL | MOV SI,1  CHKLARGEST:  CMP SI,4  JE DISPLAY    MOV BL,INPUT[SI]  CMP LARGEST,BL  JA MORE  JBE LESS  MORE:  INC SI  JMP CHKLARGEST  LESS:  MOV LARGEST,BL  INC SI  JMP CHKLARGEST    DISPLAY:  MOV AH,02H  MOV DL, LARGEST  INT 21H    EXIT:  MOV AX,4C00H  INT 21H  MAIN ENDP  END MAIN |

1. Refer to the sample output below, you are required to write a complete assembly program to let user to enter a three-character login name and a three-character password for access. If a user enters a correct login name e.g. ABC and a correct password e.g. 123, the access is granted otherwise it is denied. (PYP-04/14: 17 marks)

Marks will be awarded based on the following criteria.

|  |  |
| --- | --- |
| Assessment criteria | Marks allocated |
| Declaration of Data Items | 2 marks |
| Initialization of Data Segment | 1 mark |
| Display prompts | 1 mark |
| Receive inputs | 4 marks |
| Validate login name | 3 marks |
| Validate password | 3 marks |
| Display outputs | 2 marks |
| End program | 1 mark |
| Total: | 17 marks |

|  |
| --- |
| Sample output: |
| Login Name: ABC  Password: 123  \*\*\* Login Successfully! \*\*\*  Login Name: XYZ  Password: 123  \*\*\* Invalid Login Name. Access Denied! \*\*\*  Login Name: ABC  Password: 456  \*\*\* Invalid Password. Access Denied! \*\*\* |

|  |  |
| --- | --- |
| **Answer** | |
| .MODEL SMALL  .STACK 64  .DATA  PROMPT1 DB "LOGIN NAME:$"  PROMPT2 DB 10, 13, "PASSWORD: $"  ERRORMSG1 DB 10, 13, "\*\*\* INVALID LOGIN NAME. ACCESS DENIED! \*\*\*$"  ERRORMSG2 DB 10, 13, "\*\*\* INVALID PASSWORD. ACCESS DENIED! \*\*\*$"  VALIDMSG DB 0DH,0AH,"\*\*\* LOGIN SUCCESSFULLY \*\*\*$"  UNAME DB "ABC"  PASSW DB "123"  USERN LABEL BYTE  MAXN DB 4  ACTN DB ?  INPUTN DB 4 DUP ("$")    USERP DB 3 DUP (0)  .CODE  MAIN PROC FAR  MOV AX , @DATA  MOV DS , AX  ; ===== PROMPT AND ACCEPT  ; USERNAME AND PASSWORD ==  MOV AH , 09H  LEA DX , PROMPT1  INT 21H  MOV AH , 0AH  LEA DX , USERN  INT 21H  MOV AH , 09H  LEA DX , PROMPT2  INT 21H  MOV CX , 3  MOV BX , 0    L1: MOV AH , 01H  INT 21H    MOV USERP[BX],AL  INC BX  LOOP L1 | ; ===== AUTHENTICATE THE  ; USERNAME AND PASSWORD ===  MOV CX , 3  MOV SI , 0  L2: MOV AL , INPUTN[SI]  CMP AL, UNAME[SI]  JNE ERROR1  INC SI  LOOP L2    MOV CX , 3  MOV DI , 0  L3: MOV AL , USERP[DI]  CMP AL, PASSW[DI]  JNE ERROR2  INC DI  LOOP L3    JMP VALID  ; ===== AUTHENTICATION RESULT MESSAGE =====  ERROR1: MOV AH , 09H  LEA DX , ERRORMSG1  INT 21H  JMP EXIT  ERROR2: MOV AH , 09H  LEA DX , ERRORMSG2  INT 21H  JMP EXIT  VALID:  MOV AH , 09H  LEA DX , VALIDMSG  INT 21H    EXIT: MOV AX, 4C00H  INT 21H  MAIN ENDP  END MAIN |
|  |  |

**TUTORIAL 10: Keyboard and Screen Processing**

List out the 3 main usages of the INT 21H operations.

|  |
| --- |
| **Answer** |
| * Input * Output * Terminate processing |

What are the purposes of the following functions?

1. INT 10H function 02H

|  |
| --- |
| **Answer** |
| Set cursor |

1. INT 10H function 06H

|  |
| --- |
| **Answer** |
| Scroll screen |

1. INT 21H function 01H

|  |
| --- |
| **Answer** |
| Input byte |

1. INT 21H function 02H

|  |
| --- |
| **Answer** |
| Output byte |

1. INT 21H function 07H

|  |
| --- |
| **Answer** |
| Input byte (no echo) |

1. INT 21H function 09H

|  |
| --- |
| **Answer** |
| Output string |

1. INT 21H function 0AH

|  |
| --- |
| **Answer** |
| Input string |

Write the instructions required to set the cursor to row 12, column 24. Then rewrite the instructions set so that it involves DX register. 🡪 go to question 11

|  |
| --- |
| **Answer** |
| MOV AH, 02H ; set cursor location  MOV BH, 00 ; page number  MOV DH, 11 or 0BH ;ROW  MOV DL, 23 or 17H ;COL  INT 10H  ---Using DX register---  MOV AH, 02H  MOV BH, 00  MOV DX, 0B17H  INT 10H |

Explain the effect of the following program segment?

MOV AX , 0600H

MOV BH , 71H

MOV CX , 0000H

MOV DX , 184FH

INT 10H

|  |  |
| --- | --- |
| **Answer** | |
| MOV AX , 0600H  MOV BH , 71H  MOV CX , 0000H  MOV DX , 184FH  INT 10H | ; Clear screen and set screen scrolling  ; 1st for background color, 2nd digit for foreground color  ; start coordinate  ; end coordinate  ; Call interrupt service for screen monitoring |

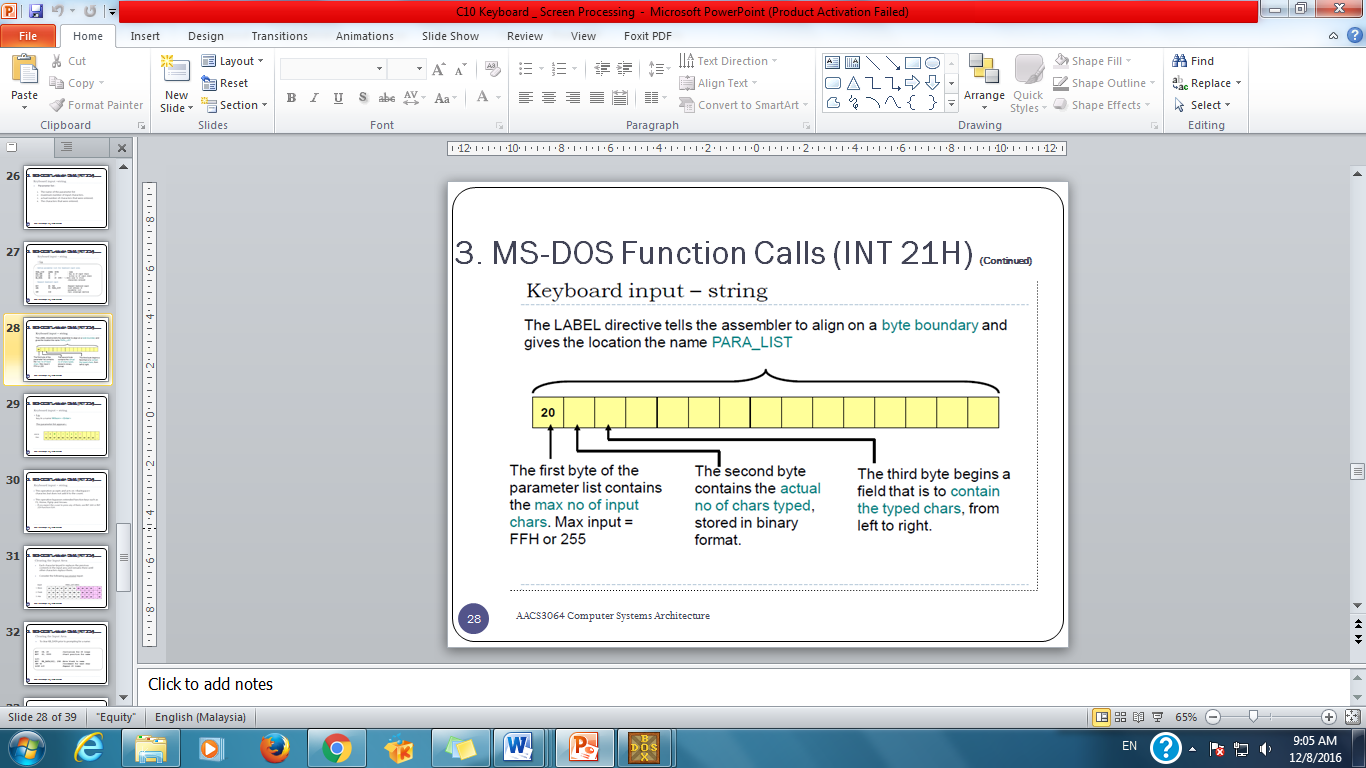
“When using INT 21H function 09H for displaying, a delimiter ($) is defined immediately following the display area.”

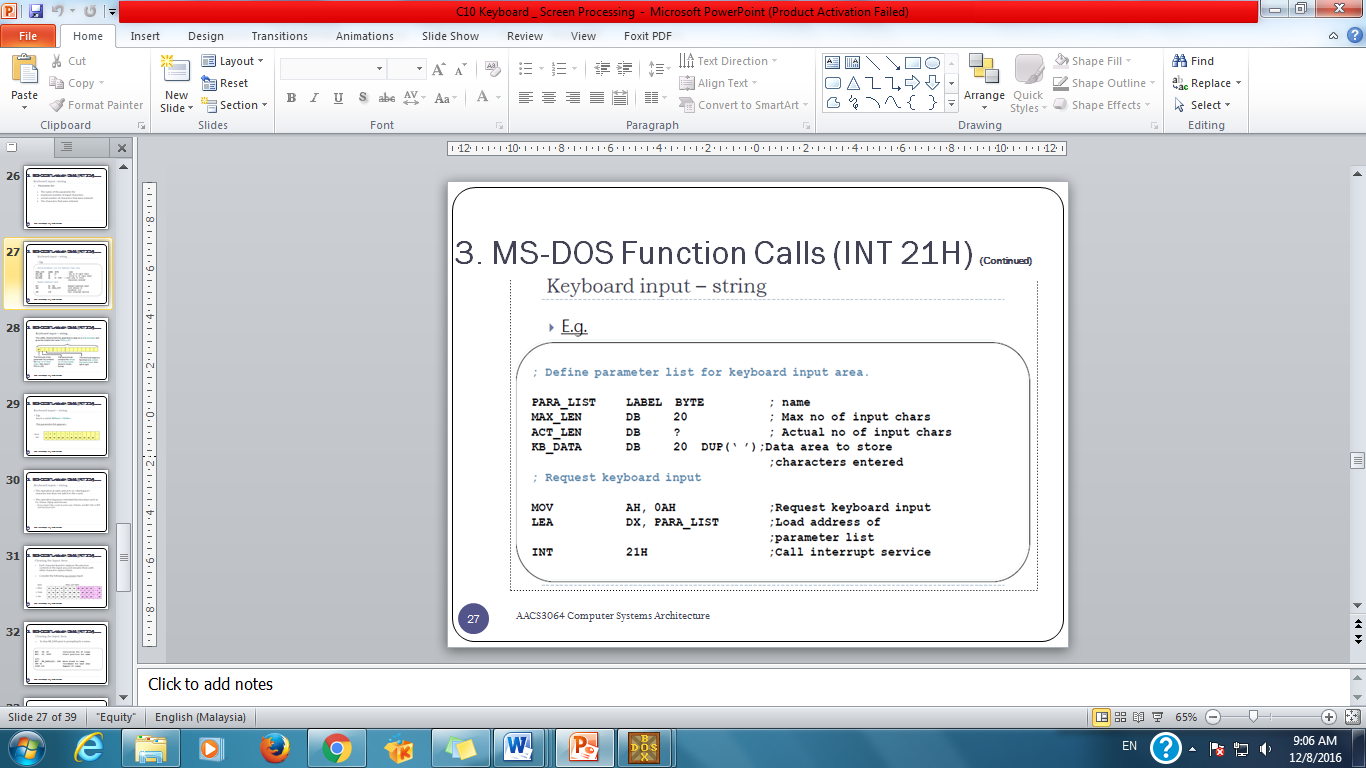
1. Why the delimiter is needed?

|  |
| --- |
| **Answer** |
| To mark the end of a string |

1. What will happen if the delimiter is missing?

|  |
| --- |
| **Answer** |
| The remaining data will be retrieved. |

INT 21H function 0AH for keyboard input requires a parameter list.



1. What will the first byte store?

|  |
| --- |
| **Answer** |
| Maximum length of the parameter |

1. What will the second byte store?

|  |
| --- |
| **Answer** |
| Actual length of the parameter |

1. What will the continuous bytes store?

|  |
| --- |
| **Answer** |
| Actual parameter data |

What register works closely with INT 21H to perform various functions?

|  |
| --- |
| **Answer** |
| AH register |

In order to display a character using INT 21H function 02H, the character to be displayed must be loaded to which register?

|  |
| --- |
| **Answer** |
| DL register |

Write instructions to reset cursor in the new line and at the left most position.

|  |
| --- |
| **Answer** |
| MOV AH, 02H  MOV DL, 0AH  INT 21H  MOV DL, 0DH  INT 21H |

For INT 21H function 09H, when LEA command is applied, where will the assembler load the effective (offset) address to?

|  |
| --- |
| **Answer** |
| DX register |

Identify the hex values of AH, DX and INT if the cursor is set at row 7, column 14 on screen using assembly language. (PYP-04/14: 3 marks)

|  |
| --- |
| **Answer** |
| MOV AH, 02H  MOV BH, 00  MOV DX, 060EH  INT 10H |

Using simplified segment directive, write an assembly language program that performs One’s Complement operation. (PYP-08/14: 15 marks)

The program should:

1. Prompt user to enter an 8-bit binary data.
2. Accept the 8-bit binary data from the user. *(Assume user will only enter 8-bit binary data)*
3. Flip the binary data accepted.
4. Display the One’s Complement result on the screen.

Marks will be awarded based on the following criteria.

|  |  |
| --- | --- |
| Assessment criteria | Marks allocated |
| Print prompts | 3 marks |
| Accept input (with proper data structure) | 2 marks |
| Flip operation using loop instruction | 8 marks |
| Print result | 2 marks |
| Total: | 15 marks |

|  |
| --- |
| Sample output: |
| Please enter 8-bit binary data >> 10100101  The One’s Complement value of the data is >> 01011010 |

|  |  |
| --- | --- |
| **Answer** | |
| .MODEL SMALL  .STACK 100  .DATA  INPUT DB 8 DUP (0)  STR1 DB "PLEASE ENTER 8-BIT BINARY DATA >> $"  STR2 DB 0DH,0AH,"THE ONE’S  COMPLEMENT VALUE OF THE DATA IS >> $"  .CODE  MAIN PROC    MOV AX,@DATA  MOV DS,AX  MOV AH,09H  LEA DX,STR1  INT 21H  MOV CX,8  MOV SI,0  GETINPUT:  MOV AH,01H  INT 21H  MOV INPUT[SI], AL  INC SI  LOOP GETINPUT  MOV SI,0  CHK:  CMP SI,7  JE DISPLAY  CMP INPUT[SI],"0"  JE CHG1  JNE CHG0 | CHG1:  MOV INPUT[SI],"1"  INC SI  JMP CHK  CHG0:  MOV INPUT[SI],"0"  INC SI  JMP CHK  DISPLAY:  MOV AH,09H  LEA DX,STR2  INT 21H  MOV CX,8  MOV SI,0  PRINTINPUT:  MOV AH,02H  MOV DL,INPUT[SI]  INT 21H  INC SI  LOOP PRINTINPUT    EXIT:MOV AX,4C00H  INT 21H  MAIN ENDP  END MAIN |

Write an assembly program that will prompt the user for input. The program will display all the numeric data input by the user, while filters all other characters. (Assume user will not enter $.)

Sample output:

|  |
| --- |
| Please enter any data and press enter >> 12A3%C&4  The numerical value is >> 1234 |

|  |  |
| --- | --- |
| **Answer** | |
| .MODEL SMALL  .STACK 64  .DATA  S1 DB "PLEASE ENTER ANY DATA AND PRESS ENTER>> $"  S2 DB 0DH,0AH,"THE NUMERICAL VALUE IS >> $"  ARRAY LABEL BYTE  MAX DB 100  ACT DB 0  ARRAYDATA DB 100 DUP ("$")  .CODE  MAIN PROC  MOV AX,@DATA  MOV DS,AX  ;----OUTPUT STR  MOV AH,09H  LEA DX,S1  INT 21H  ;--INPUT STR  MOV AH,0AH  LEA DX,ARRAY  INT 21H  ;----OUTPUT STR2  MOV AH,09H  LEA DX,S2  INT 21H | ;---CHECKNUMBER  MOV SI,0  CHKNUM:  CMP ARRAYDATA[SI],"$"  JE EXIT  CMP ARRAYDATA[SI],"0"  JAE CHK2  JB CHKNEXT  CHK2:  CMP ARRAYDATA[SI],"9"  JBE PRINTNUM  JA CHKNEXT  CHKNEXT:  INC SI  JMP CHKNUM  PRINTNUM:  MOV AH,02  MOV DL,ARRAYDATA[SI]  INT 21H  INC SI  JMP CHKNUM  EXIT:  MOV AX,4C00H  INT 21H  MAIN ENDP  END MAIN |

**TUTORIAL 11: Input and Output Facilities**

1. Define I/O module. Discuss the roles of I/O module in CPU – I/O communication.

(PYP-01/14: 2, 8 marks)

|  |
| --- |
| **Answer** |
| I/O module   * It serves as an interface between the CPU and the specific device. * It accepts commands from the CPU on one side and control the device on the other. * Roles: * Recognizes messages addressed to it and accepts commands from the CPU * Provides a buffer where the data from memory can be held until it can be transferred to the disk. * Provides the necessary registers and controls to perform a direct memory transfer. * Provides interrupt capability to notify the CPU when the operation is completed. |

1. Differentiate the following I/O techniques: Programmed I/O and Direct Memory Access (DMA) in term of their characteristics. (PYP-01/14: 10 marks)

|  |  |  |
| --- | --- | --- |
| **Answer** | | |
|  | P I/O | DMA |
| Devices involved | Slow devices | Fast devices |
| Data size involved | Single I/O | Block of data |
| Machine cycle | Full | Initiation & notification |
| CPU involvement | Full | Partial |
| Data required | * Address * Data | * Location of data in device * Location of data in memory * Data size * Direction of transfer |
| Requirement | Direct connection | * I/O module & memory are connected * I/O modules are capable to read / write * Conflict between I/O module & CPU must be avoided |

1. An I/O technique is known as Interrupt-driven I/O. Explain the technique. (PYP-04/14: 5 marks)

|  |
| --- |
| **Answer** |
| Interrupt-driven I/O   * Signals that causes the CPU to alter its normal flow on instruction execution. * Advantages: - * Free CPU from waiting for events * Provides control for external input * Examples: - * An unexpected user input * Illegal instructions * Multitasking, multiprocessing * An abnormal situation (paper jam, out of paper, bad sector) |

1. Explain the FOUR (4) usages of Interrupt. (PYP-08/14: 8 marks)

|  |
| --- |
| **Answer** |
| * As an external event notifier * As a completion signal * As a mean of allocation of CPU time * As abnormal event indicator |

1. How does Direct Memory Access (DMA) release the waiting time of CPU? (PYP-04/14: 3 marks)

|  |
| --- |
| **Answer** |
| * The transfer is initiated by a program in the CPU by using PIO, but the CPU can then bypass for the remainder of the transfer. * The I/O module will notify the CPU with an interrupt when the transfer is completed. * Once this has occurred, the data is in memory, ready for the program to use. |

1. Interrupt is a signal sent to the CPU to indicate an event that requires immediate attention. Servicing the interrupt is a process where the CPU branches to the interrupt handler program from its current program. Explain the flow of servicing an interrupt in detail. (PYP-08/13: 4 marks)

|  |
| --- |
| **Answer** |
| 1. Suspends program in progress 2. Save context, including last instruction executed and data values in registers, in PCB (process control block). 3. Branches to interrupt handler program (interrupt routine) 4. Resume control to the interrupted program by referring to the PCB. |

1. Suppose you are sending a block of data from a disk to memory.
2. Identify the most appropriate I/O handling technique to support the data transfer process as mentioned. Explain your choice. (PYP-08/13: 2 marks)

|  |
| --- |
| **Answer** |
| Direct memory access (DMA) |

1. Based on the scenario given in Q7, explain the process of the I/O handling technique that you mentioned in Q7 (a) with the aid of an appropriate diagram. (PYP-04/14: 6 marks)

|  |
| --- |
| **Answer** |
| 1. PIO is used to prepare I/O module for transfer by providing required info & initiating transfer. 2. DMA transfer. In this case, data is transferred from disk to memory. 3. Upon completion, disk controllers sends completion interrupt to CPU.  |  |  |  |  |  | | --- | --- | --- | --- | --- | | CPU |  | I/O module  (disk controller |  | Disk | | Memory |  |  |  |  | |

1. Distinguish between a bus architecture and a channel architecture in terms of their characteristics and operations. (PYP-08/12: 6 marks)

|  |  |  |
| --- | --- | --- |
| **Answer** | | |
| I/o architecture | Bus architecture | Channel architecture – for IBM mainframe computer |
| Nature | Simple | Complex |
| Application | PC | Mainframe |
| Components | Backbone + interface | Channel subsystem (I/O processor) |
| Pros | Easy expand & flexible access | Better control & free CPU |
| Cons | Loose management | Max 8 channel path |
| Diagram |  |  |

1. Four pieces of data must be provided to the I/O controller for a particular I/O device to initiate a Direct Memory Access (DMA) transfer. What are they? (PYP-04/10: 4 marks)

|  |
| --- |
| **Answer** |
| 1. The location of data on the I/O device 2. The starting location of the block of data in memory 3. The size of block to be transferred 4. The direction of transfer.   Read (I/O memory)  write (memory I/O) |

1. In the event of multiple interrupts, illustrate how these multiple interrupts could be handled by using vectored interrupt and polled interrupt respectively. Support your answer with a diagram for each.

|  |  |
| --- | --- |
| **Answer** | |
| Vectored interrupt | Polled interrupt |
| The address of the interrupting device is included as part of the interrupt. | The general interrupt that shared by all devices. Identifies the interrupting device by polling each device.  *(check device one by one)* |
|  |  |

(PYP-01/12: 10 marks)

1. Consider the interrupt that occurs at the completion of a disk transfer (Disk to Memory).
2. “Who” is interrupting “Whom”?

|  |
| --- |
| **Answer** |
| Disk is interrupting CPU. |

1. Why is the interrupt used in this case? What would be happened if there were no interrupt capability on this computer?

|  |
| --- |
| **Answer** |
| To notify the completion of transfer.  The resources cannot be released to the others. |

**TUTORIAL 12: Computer Architecture**

1. Briefly explain how a system performance is affected.

|  |
| --- |
| **Answer** |
| Factors affecting the performance of a system are including the following:   * Multiple CPUs * Faster system clock speed * Wider instruction and data path * Faster disk access * Memory access time * Increase amounts of memory and faster memory   Briefly explain for each of them. |

1. What is the main different between loosely-coupled system and parallel system?

(PYP-01/10: 6 marks)

|  |
| --- |
| **Answer** |
| 1. Distributed Operating systems are also referred to as Loosely Coupled systems whereas parallel processing systems are referred to as tightly coupled systems. 2. A Loosely coupled system is one in which the processors do not share memory and each processor has its own local memory whereas in a tightly coupled system there is a single system wide primary memory shared by all the processors. 3. The processors of distributed operating systems can be placed far away from each other to cover a wider geographic area which is not the case with parallel processing systems. 4. The no. of processors that can be usefully deployed is very small in a parallel processing operating system whereas for a distributed operating system a larger no. of processors can be usefully deployed. 5. In the distributed operating system there is an unpredictable communication delays between processors whereas the processors in the parallel processing system share over an interconnection network |

1. Cache memory is used as a technique to enhance memory performance.
2. With the aid of a diagram, explain how a cache memory helps to enhance memory performance.

(PYP-08/13: 5 marks)

|  |
| --- |
| **Answer** |
| Cache memory   * Definition: a small amount of high speed memory between CPU & main storage, which is invisible to programmer and cannot be directly addressed by CPU. * Diagram:   CPU  Memory  Cache memory |

1. What will happen when there is a cache miss? (PYP-08/13: 2 marks)

|  |
| --- |
| **Answer** |
| * If there is a miss, a miss requires the cache controller to select a line for replacement from memory. |

1. Analyze the impact of cache miss on the execution of instruction. (PYP-08/13: 4 marks)

|  |
| --- |
| **Answer** |
| * Data not found in cache. Processor loads data from Memory and copies into cache. This results in extra delay. |

1. Explain what will happen when a cache memory is full. (PYP-08/15: 4 marks)

|  |
| --- |
| **Answer** |
| * When cache memory is full, some block in cache memory must be selected for replacement, usually based on a least recently used basis. |

1. List and explain TWO (2) configurations of a multiprocessing system. (PYP-08/13: 6 marks)

|  |
| --- |
| **Answer** |
| Two ways of configuring a multiprocessing system are:   * Tightly coupled systems   + Also called multiprocessor systems   + Characteristics: identical access to programs, data, shared memory, I/O, etc. * Loosely coupled systems   + Also called cluster / multi-computer systems   + Characteristics: each system has its own CPU, memory and I/O facilities. |

1. Explain the following techniques in enhancing memory performance.
2. Wider path for memory access

|  |
| --- |
| **Answer** |
| * Retrieve multiple bytes instead of 1 byte at a time * Several bytes / words can be read / written between CPU and memory with each access. |

1. Cache memory (PYP-04/13: 4 marks)

|  |
| --- |
| **Answer** |
| * A small amount of high speed memory between CPU and main memory storage * It is invisible to the programmer and cannot be directly addressed in the usual way. |

1. What performance improvement is offered by the use of memory interleaving?

|  |
| --- |
| **Answer** |
| Performance improvement offered by memory interleaving:   * Increase the effective rate of memory access by dividing memory into parts * Allow memory access to more than one location at a time. |

1. Explain how pipelining serves to reduce the average number of steps in the execution part of the fetch-execute cycle.

|  |
| --- |
| **Answer** |
| The pipelining   * Definition : It overlaps instruction so that more than one instruction is being worked on at a time. * Process : Each instruction completes a step, the following instruction moved into the stage jus vacated. Thus, when the first instruction is completed, the next instruction is already one stage short of completion.  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  |  | Fetch | Decode | Execute | Store |  | |  | Fetch | Decode | Execute | Store |  |  | | Fetch | Decode | Execute | Store |  |  |  |   Instruction3  Instruction2  Instruction1    Time 0 1 2 3 4 5 |

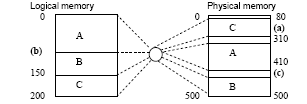
1. The designer of a new operating system has proposed the use of virtual storage memory management for real time processing so that the system can handle programs that are too large to fit in the limited memory space. What are the implications of this decision in term of the way that virtual storage works?

|  |
| --- |
| Answer |
| Virtual storage   * Only the active pages require corresponding frames in the page table and in physical memory. * Load only a small part of a program and have it executed. * Process:   + In the case the two requirements met, the instruction will be executed. * The instruction / data must be in physical memory. * The page table for the program must contain an entry that maps the virtual address being accessed to the physical location containing the instruction / data.   + In the case of not enough frames available to populate the page table when the program is loaded, load only a small part of a program and have it execute. |

1. Differentiate between logical address and physical address.

|  |
| --- |
| **Answer** |
| * Physical addresses do not need to be consecutive. * Logical addresses mapped to physical addresses. |

1. Complete the diagram below.



|  |
| --- |
| **Answer** |
| (a) 130  (b) 100  (c) 450 |