**TUTORIAL 1: Numbering Systems**

1. Convert the following to decimal numbers.
2. 111001112
3. 55778
4. ABCD H
5. Convert the followings to Base-2, Base-8 and Base-16 respectively.
6. 3064 D
7. 201610
8. 2899 base-10
9. Convert the following numbers to the respective base.
10. 11111010001111002 to base-8 and base-16 respectively
11. 11000011101001012 to base-8 and base-16 respectively
12. 1EFH to base-2 and base-8 respectively
13. 257ACH to base-2 and base-8 respectively
14. 7548 to base-16
15. 44716 to base-8
16. 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)

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

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

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

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

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)

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

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

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

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

1. Perform the following operations and show the answers in the respective number base. You are required to show your working steps clearly
2. ABC16 + FFF16 (PYP-01/14: 3 marks)
3. 1258 – 778 (PYP-01/14: 2 marks)
4. 10112 x 1012
5. 110112 x 10112
6. 100111002 + 010010112
7. Perform the following operations and show the answers in the respective number base. You are required to show your working steps clearly
8. 4618 + 5158
9. 2247 - 1367
10. 173H x AAH
11. 111110102 + 11108 (Show your final answer in Hex format.) (PYP-04/13: 3 marks)
12. 306416 x 2138 (Show your final answer in Hex format.) (PYP-08/12: 3 marks)

**TUTORIAL 2: Numerical Data Representation**

**Section A: Signed Number**

1. Under what circumstances the Two’s Complement is used?
2. Convert the 8-bit binary number 11010111 into decimal number if the binary number is a(n):
3. Unsigned number (PYP-04/14: 1 mark)
4. Signed number (PYP-04/14: 2 marks)

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

|  |  |  |
| --- | --- | --- |
| Flag | Carry | Overflow |
| Definition |  |  |
| Detect in signed or unsigned number? |  |  |
| How to detect? |  |  |
| Example |  |  |

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)

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)

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)
2. Verify your answer by showing the answer in signed decimal value. (PYP-08/13: 3 marks)
3. Justify the validity of the answer obtained. (PYP-08/13: 1 mark)
4. Does overflow occur? Justify your answer. (PYP-08/13: 1 mark)
5. Assuming an 8-bit system is involve.
6. Solve the following operation using Two’s Complement method: (PYP-08/14: 5 marks)

(-910) + (-810)

1. Verify your answer by showing the answer in signed decimal value. (PYP-08/14: 1 mark)
2. Justify the validity of the answer obtained. (PYP-08/14: 4 marks)
3. Does overflow or/and carry occur?

**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)
3. 123.1235 to Decimal (PYP-08/14: 2 marks)
4. 100100011111.112 to Octal (PYP-08/14: 2 marks)
5. Perform the following operations. Show your working steps clearly. If the operation is illogical, explain the reason.
6. Convert 6A.9610 to hexadecimal number
7. Convert 1807.6510 into a hexadecimal number (PYP-08/13: 2 marks)
8. Convert 101011.01112 into a decimal number (PYP-04/13: 2 marks)
9. 111100110011.110000012 + 20.510. Show your answer in Hex format. (PYP-08/12: 3 marks)
10. 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)
2. Convert 55220311 to scientific notation.
3. Convert 95575321 to scientific notation.
4. Convert 30.81510 to the SEEMMMMM format.
5. 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.
6. Add these two numbers. Present your result in standard decimal sign-and-magnitude notation.

55020311

15375321 (PYP-08/13: 5 marks)

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

15176323

15485496 (PYP-08/13: 5 marks)

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

55152295

15256608 (PYP-04/14: 5 marks)

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)
2. Multiply these two numbers. Show the result in sign-magnitude notation. (PYP-08/12: 3 marks)
3. 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)
4. 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)
5. 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)
6. 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)

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

1. Explain each of the following terms:
2. Segment: Place to store stack information
3. Offset: The relative distance from one point. Usually refers to the number of bytes away.
4. Register: High-speed, temporary memory.
5. Explain the purpose of the following items:
6. Code segment: Stores machine instruction (PYP-01/16: 1 mark)
7. Code segment (CS) register: Stores the address to the code segment of the memory (PYP-01/16: 1 mark)
8. Instruction pointer (IP) register: Points to the next instruction to be executed (PYP-01/16: 1 mark)
9. Accumulator (AX): Used in input/output, and most arithmetic operations
10. Count register (CX): Used in arithmetic operations, and looping
11. Parity flag (PF): Used for checksum and error-checking
12. (a) List and explain FOUR (4) main segment in a CPU: (PYP-04/14: 4 marks)
    1. Code segment (CS): Stores machine code
    2. Stack segment (SS): Stores defined variables and constants
    3. Data segment (DS): Stores local function variable & function parameters
    4. Extra segment (ES): Spare segment, used to specify location in memory

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

|  |  |  |
| --- | --- | --- |
| Registers |  | RAM |
| **Similarities** | | |
| Both stores data used to execute programs | | |
| **Differences** | | |
| Using address of register | **Access Method** | Using name to reference |
| In CPU | **Location** | On Motherboard |
| In sequence | **Content order** | May not be in sequence |

1. Differentiate between a segment:offset address and an absolute address.
   1. **Absolute address** directly refers to a particular location.
   2. **Segment:offset** address needs to be calculated to find the location intended.
2. 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.
3. 738 + 258 (PYP-08/11: 4 marks)
   1. 1208
   2. To binary: 0 0101 0000
   3. In Hex: ­5016
   4. Stored in AX:

|  |  |
| --- | --- |
| AX | |
| AH | AL |
| 00 | 50 |

1. 11112 x 1112 x 112  (PYP-08/11: 4 marks)
   1. Decimal: 15 \* 7 \* 3 = 525
   2. Hex: 20D
   3. Stored in AX:

|  |  |
| --- | --- |
| AX | |
| AH | AL |
| 2 | 0D |

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.
   1. CS:IP
   2. Next: 02B3 \* 10H + 25H = 02B55H
2. Calculate the corresponding 20-bit absolute memory address using the SS:SP.
   1. 09AF \* 10H + 94H = 9AF0H + 94H = 09B84H

(PYP-08/13: 4 marks)

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)
   1. 12AB \* 10H + 0020H = 12AB0 + 0020 = 12AD0H
   2. 20-bit binary: 01 0010 1010 1100 0000
2. Complete the following table.

|  |  |  |  |
| --- | --- | --- | --- |
| Register | Definition | High speed, temporary, data storage location inside CPU | |
| Characteristics | Temporary, small, fast. | |
| **Types of register** | **General purpose** | **Address** | **Status** |
| **Aim** | Arithmetic, data storing & movement. | Ind add0 | Indicate state |
| **Size** | 8 16 | 16 | 1 |
| **Examples** | 16: a b c d x  8 hl | Seg: cs | of |

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.
   1. ZF
3. Used for error checking when there is a possibility that data might be altered or corrupted.
   1. PF
4. When the result of an unsigned arithmetic operation is too large to fit into the destination.
   1. CF (OF only for sign)
5. Used for arithmetic and data movement.
   1. AX….DX
6. Used for counting loops.
   1. CX
7. 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 NC, PL, NZ, NV

ADD AL, AAH AX = 61CA ; NC, PL, NZ, NV

ADD AH, FFH AX = 1(60CA) ; CY, PL, NZ, NV

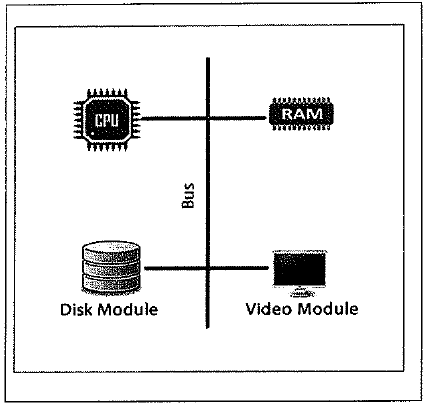
ADD AX, 2 AX = 60CC; CY, PL, NZ, NV

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Set** | **Flag mnemonics** | **Clear** | **Flag mnemonics** |
| Overflow | OV | No overflow | NV |
| Direction down | DN | Direction up | DU |
| 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)
   1. Bus is a physical connection to transfer data inside a computer system. Bus contain electrical conductors called “lines”, and carrier computer signals.
2. Name the bus configuration as shown in the Figure 1 above. (PYP-01/16: 2 marks)
   1. Multipoint bus
3. List and explain THREE (3) possible types of bus line that can be found in a multipoint bus configuration. (PYP-01/16: 6 marks)
   1. **Data bus**. Transports data, bidirectional.
   2. **Address bus.** Specifies data recipient, identifies source & recipient of data on data bus.
   3. **Control bus.** Provide control for synchronization, and control of bus & modules connected.
4. 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)
   1. Point-to-point bus has separate bus lanes for each pair of connecting modules. Multipoint bus has bus lanes that are connected to more than 2 modules.
   2. Point-to-point bus sends and receive data directly. Multipoint bus “broadcasts” data, and only the correct recipient “accepts” the data.
5. *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)

* + - MAR holds the address of data inside the RAM to be opened for data access by the CPU.
    - MDR stores a copy of data pointed by MAR for read access by the CPU.
    - MDR stores data in memory for write-access by CPU.

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

(PYP-08/12: 3 marks)

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

(PYP-08/12: 4 marks)

1. One large modern computer has a 36-bit MAR. How much memory can this computer address?
2. 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.

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

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

**TUTORIAL 5: Machine Execution**

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

**Label, Mnemonic, Operand, Comment**

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

1. Memory as operand
   1. Segment:Offset
   2. MOV AX, NUM
   3. MOV NUM, AX (register to memory)
   4. MOV NUM, DIGIT (CANNOT variable to variable)
2. Register as operand
   1. Registers must be same type. 8-bit cannot mix with 16-bit register.
   2. MOV AX, BX
   3. MOV AH, BH
   4. MOV AX, BH (Cannot)
3. Immediate value as operand
   1. For 2-operand instruction, only as 2nd operand (first must not be immediate value)
   2. MOV AX, 10
   3. MOV NUM (variable), 10
   4. MOV 30, BX (Cannot)
      * 1. Issue DEBUG commands for the following operations: (POPULAR QUESTION)
4. Reset current IP value to 200. (PYP-08/14: 2 marks)
   1. -r ip
   2. 200
5. Insert the machine instruction 10 AB 2A 35 and 5D into Code Segment with Offset Address 100. (PYP-08/14: 2 marks)
   1. -e cs:100 10 ab 2a 35 5d
6. Translate the machine instruction from Offset Address 100 to 108 into assembly code.
   1. -u 100 108

(PYP-08/14: 2 marks)

1. Execute a group of 5 instruction starts from the offset address 100H. (PYP-08/11: 2 marks)
   1. -p=100, 5
2. 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)
   1. -c 100 10A 150
      * 1. Explain the following DEBUG operations.
3. –D CS:200 (PYP-01/14: 1 mark)
   1. Dump the first 128-bytes machine code in hex notation from CS offset 200.
4. –A 200

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

* 1. Assemble the code SUB BL,42 into CS offset 200.

1. –T 3 (PYP-04/14: 1 mark)
   1. Trace execution of 3 instructions from the current offset.
2. H 3064 2130
   1. Calculate the hex sum and difference of 3064H and 2130H.

* + - 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
   1. MOV AX, 4269
   2. 
2. Move the 7012H to BX register
   1. MOV BX, 7012
   2. 
3. Add the value of BX to AX register
   1. ADD AX, BX
   2. 
4. Exchange the content in AX and BX
   1. XCHG AX, BX
   2. 
      * 1. Describe each of the following DEBUG commands. Complete the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| Command | Purpose | Format / E.g. | Output |
| A | Convert symbolic code to object code | A [address] | Object codes. (Not shown.) |
| C | Compare 2 groups of content in memory segment | C range address | Segment:offset addresses & content of the selected groups |
| D | Display the content of memory segment | D [range] | 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 range list | Not shown.  Use D command to confirm filled in. |
| H | Perform hexadecimal computation | H value1 value2 | Result of addition and subtraction |
| P | Proceed to execute a group of instruction | P [=address] [number] | Changes of registers & next instruction |
| Q | Quit the DEBUG program | -Q | Back to root directory |
| R | Retrieve / edit the content of register(s) | R [register] | Content of register(s) |
| T | Trace the execution of instruction 1 by 1 | -T | Changes of registers & next instruction |
| U | Convert object code to symbolic code | U [range] | 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.
   1. -e cs:101 54
2. Write the command to change the instruction pointer address to 200H.
   1. -r ip 200
      * 1. Assume that the IP register has value 0100H. What is its value after the instruction B8 25 00 is executed?
           1. 0103H

**TUTORIAL 6: Computer Architecture**

1. Write the command line to perform the followings: (FINAL WON’T ASK)
2. To assemble a source program named CSA.ASM into listing and object files.
   1. masm csa.asm
3. To link an object file named CSA.OBJ into executable and map files.
   1. Link csa.obj
4. To directly execute CSA.EXE from DOS.
   1. Csa.exe
5. To execute CSA.EXE through DEBUG program.
   1. TD csa.exe
6. Briefly describe the function of TWO-PASS ASSEMBLER in the process of assembling an Assembly Language program. (PYP-08/12: 3 marks)
   1. Resolve forward referencing. Two-pass assembler constructs the symbol table on the first pass, and uses the symbol table to complete object code for instructions on the second pass.
7. Illustrate the process of editing, assembling, linking and executing assembly language programs in an appropriate diagram
   1. Editing (User) [.ASM] 🡪 Assembling (Assembler) 🡪 Linking (Linker) 🡪 Executing (CPU)
   2. Copy paste diagram
8. Write an assembly program using simplified segment definitions to perform the following operations: (NOTE: NEED TO INCLUDE SKELETON, AKA ENTIRE TEMPLATE).
9. Move the immediate value hex 40 to the AL register
   1. MOV AL, 40
10. Shift AL contents one bit left
    1. SHL AL, 1
11. Move immediate value hex 1A to BL
    1. MOV BL, 1A
12. Multiply AL by BL
    1. MUL BL
13. Modify your answer in Q4 above for the following requirements:
14. Define a 1-byte item named ITEMA containing hex 40 and another named ITEMB containing hex 1A.
    1. ITEMA DB 40
    2. ITEMB DB 1AH
15. Define a 2-byte item named ITEMC with no constant
    1. ITEMC DW ?
16. Move the contents of ITEMA to AL and shift left one bit
    1. MOV AL, ITEMA
    2. SHL AL, 1
17. Multiply AL by ITEMB
    1. MOV BL, ITEMB
    2. MUL AL
18. Move the product in AX to ITEMC
    1. MOV ITEMC, AX

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

**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)
2. A 1-byte item containing the hex equivalent to decimal 40.
   1. DATA1 DB 40D
3. A 2-byte item containing an undefined value.
   1. DATA2 DW ?
4. A string with the characters H,E,L,L,O
   1. DATA3 DB “HELLO$”
5. An item containing the consecutive word values 1, 3, 5, 7 and 9.
   1. DATA4 1,3,5,7,9
6. 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)
7. 2NAME
   1. INVALID. First character cannot be digits
8. Total\_1
   1. VALID
9. #TelNo
   1. VALID
10. .Counter
    1. INVALID. First character cannot be `.`
11. @Word
    1. INVALID. Used in @data and other special identifiers.
12. 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 |
| Errors:  Line 5  Line 9  Line 12 (?) Line 13 Line 14 Line 16  Line 17 | DATA2 DW 380 (DB max size = 255), reduce or change  MOV AX, @DATA  MOVZX  MOV AX, DATA3  MOV DX, 4C00H  MAIN ENDP  END MAIN |

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

MOV AX, 0

MOV AL, 34

MOV BL, 04

MOV CL, AL

DIV CL

MUL BL

SHL AL, 1

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

|  |  |  |  |
| --- | --- | --- | --- |
| AX | | BL | CL |
| AH | AL |
| 00 | 00 |  |  |
| 00 | 34 |  |  |
| 00 | 34 | 04 |  |
| 00 | 34 | 04 | 34 |
| 00 | 01 | 04 | 34 |
| 00 | 04 | 04 | 34 |
| 00 | 08 | 04 | 34 |
| 00 | 07 | 04 | 34 |

1. Consider the cde 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= 4 , BX = 32  SHR BL, 1 ; AX= 4 , BX = 19  ADD BL, DATA2 ; AX= 4, BX = 2B  MUL AL ; AX= 10 , BX = 2B (Note: 4\*4 = 16D = 1H)  SUB BX, AX ; AX= 10 , BX = 1B |

NOTE: All in hexadecimal

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| .DATA  DATA1 DB 32H  DATA2 DB 12H  .CODE | AX | | BX | |
| AH | AL | BH | BL |
| MOV AX, 0004H | 00 | 04 |  |  |
| MOV BX, 0000H | 00 | 04 | 00 | 00 |
| MOV BL, DATA1 | 00 | 04 | 00 | 32 |
| SHR BL, 1 | 00 | 04 | 00 | 19 |
| ADD BL, DATA2 | 00 | 04 | 00 | 2B |
| MUL AL | 00 | 10 | 00 | 2B |
| SUB BX, AX | 00 | 10 | 00 | 1B |

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
   1. MOV AX, A
   2. SUB B, AX
3. C = A + B
   1. MOV AX, A
   2. ADD AX, B
   3. ADD C, AX
4. A = - (A + 1)
   1. INC A
   2. DEC A
5. B = 4 \* B + 6
   1. MOV AX, B
   2. ADD AX, AX
   3. ADD AX, AX
   4. ADD AX, 6
   5. MOV B, AX
6. A = B - A – 1
   1. MOV AX, B
   2. SUB AX, A
   3. DEC AX
   4. MOV A, AX
7. 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 |

**TUTORIAL 8: Assembly Language Fundamentals II**

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

|  |  |  |  |
| --- | --- | --- | --- |
|  | Short | Near | Far |
| Distance covered | -128🡪127 | -32768🡪32767 | Another segment |
| Instruction(s) used | JMP, Jnnn LOOP | JMP | 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 –

; load into DI

LEA DI, input

MOV CX, 3

L1:

; loop

; ask for input

MOV AH, 01h

INT 21h

; store into array

MOV [DI], AL

; increment DI

INC DI

LOOP l1

OR

MOV SI,0

MOV CX, 3

L1:

MOV AH, 01H

INT 21H

MOV INPUT[SI], AL

INC SI

LOOP L1

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, "$"  ----------------------------------------------------------------------- << ERROR, semicolon ‘;’  .CODE  MAIN PROC FAR  MOV AX, DATA << ERROR 1, @DATA  MOV DX, AX << ERROR 2, DS  MOV CX, 9  L1: MOV AH, 09H  LEA DS, MESSAGE1 << ERROR 3, DX  INT 21H  LOOP L1  MOV AX, 4C00H  INT 21H  MAIN END << MAIN ENDP  END MAIN |

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

(PYP-01/14: 5 marks)

1. How many times will the message “HI ASSEMBLY!” be displayed on the screen when the program runs? (PYP-01/14: 1 mark)
   1. **9 times**
2. 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)

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

L1 :

….

LOOP L1

**1 times**

1. L2 : MOV CX , 10

…..

LOOP L2

**Endless loop**

1. MOV CX , 10

L3 :

INC CX

LOOP L3

**Endless Loop**

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AL | OF | ZF | SF | CF |
| 1111 1111 | NV | NZ | NG | NC |
| 0000 0000 | NV | NZ | PL | CY |

1. MOV BL , 24H

SUB BL , BL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BL | OF | ZF | SF | CF |
| 0010 0100 | NV | NZ | PL | NC |
| 0000 0000 | NV | ZR | PL | NC |

1. MOV CL , 10101010B

ADD CL , 01010101B

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CL | OF | ZF | SF | CF |
| 1010 1010 | NV | NZ | NG | NC |
| 1111 1111 | NV | NZ | NG | NC |

1. MOV DL, 11001100B

ADD DL, 01110011B

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DL | OF | ZF | SF | CF |
| 1100 1100 | NV | NZ | NG | NC |
| 0011 1111 | 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"

LEA DI, char\_string

MOV CX, 10

L1:

; loop

MOV BL, [DI]

ADD BL, 20h

MOV [DI], BL

INC DI

LOOP l1

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 = 1
2. MOV DL, [BX + 3] ; DL = 4
3. MOV AL, [BX + DI] ; AL = 5

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

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

.MODEL SMALL

.STACK 100

.DATA

tmp DB ?

times DW 3

nl DB 13, 10, "$"

.CODE

MAIN PROC

MOV AX,@DATA

MOV DS,AX

MOV CX, times

; outer loop : 3 times

L1:

MOV tmp, CL

MOV CX, times

; inner loop (3,2,1)

l2:

MOV AH, 2

; print star

MOV DL, "\*"

INT 21h

LOOP l2

MOV AH, 9

LEA DX, nl

INT 21h

DEC times

MOV CX, tmp

LOOP l1

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 |

.MODEL SMALL

.STACK 100

.DATA

tmp DB ?

times DW 3

num DB "1"

nl DB 13, 10, "$"

.CODE

MAIN PROC

MOV AX,@DATA

MOV DS,AX

MOV CX, times

; outer loop : 3 times

L1:

MOV tmp, CL

MOV CX, times

; inner loop (3,2,1)

l2:

MOV AH, 2

; print star

MOV DL, num

INC num

INT 21h

LOOP l2

MOV AH, 9

LEA DX, nl

INT 21h

DEC times

MOV CX, tmp

LOOP l1

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 |

.MODEL SMALL

.STACK 100

.DATA

tmp DW ?

times DW 5

num DB 1

nl DB 13, 10, "$"

.CODE

MAIN PROC

MOV AX,@DATA

MOV DS,AX

MOV CX, times

; outer loop : 3 times

L1:

MOV tmp, CX

MOV CX, times

; inner loop (3,2,1)

MOV BL, times

l2:

MOV AH, 2

; print star

MOV BL, num

MOV DL, BL

ADD DL, '0'

INC num

INT 21h

LOOP l2

MOV AH, 9

LEA DX, nl

INT 21h

; reset num

MOV num, 1

DEC times

MOV CX, tmp

LOOP l1

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.
   1. Answerer’s note: Can use any label.
2. Is CL equal to or smaller than DL?
   1. CMP CL, DL
   2. JLE L1
3. Is AL equal to or smaller than BL?
   1. CMP AL, BL
   2. JBE L2
4. Is AL greater than BL?
   1. CMP AL, BL
   2. JA L3
5. Is CL greater than DL?
   1. CMP CL, BL
   2. JG L4
6. Does DL contain zero?
   1. CMP DL, 0
   2. JZ L5
7. Is there an overflow?
   1. JO L6
8. Assume that DL contains 01111001 and that an item named BOOL\_AMT contains 11100011. Determine the effect on DL for the following unrelated operations:
9. AND DL , BOOL\_AMT
   1. 01100001
10. OR DL , BOOL\_AMT
    1. 11111011
11. XOR DL , BOOL\_AMT
    1. 1001010
12. AND DL , 00000000B
    1. 00000000
13. XOR DL , 11111111B
    1. 10000110
14. 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.
    1. AND AL, 1100B (REMEMBER TO PUT B)
    2. JZ L3 (Can straight away jump because check AL)
    3. JNZ L4
    4. Alternate answer: OR, can use SHR and SHL
15. Translate the following C statements into assembly language. (FAVOURITE QUESTION)
16. if (num1 == num2)

{

X = 1; Y = 2;

}

ANSWER:

MOV BL, num1

CMP BL, num2

JE L1

JMP FINISH

L1:

MOV X, 1

MOV Y, 2

ALTERNATE ANSWER:

JNE FINISH

MOV X,1

MOV Y,2

FINISH:

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

{

X = 1;

}

CMP AL, BL

JLE FINISH

CMP BL, CL

JLE FINISH

MOV X, 1

FINISH:

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

{

X = 1;

}

CMP AL, BL

JG L1

CMP BL, CL

JG L1

JMP FINISH

L1:

MOV X,1

FINISH:

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.

Answer (only cover a and e):

CHAR DB “This is my favorite”

MOV CX, 19

MOV SI, 0

L1:

CMP CHAR[SI], ‘a’

JE CA

CMP CHAR[SI], ‘e’

JE CE

JMP CONTINUE

CA: INC countA

JMP CONTINUE

CE: INC countE

CONTINUE:

LOOP L1

; After that just display

Sample output:

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

NOTE: HOT Topic in Exam

NUM DB 7,5,9,8,2,4

MOV CX, 6

MOV SI, 0

L1:

CMP NUM[SI], 5

JLE L2

MOV AH, 02H

MOV DL, NUM[SI]

ADD DL, 30H

INT 21H

L2:

INC SI

LOOP L1

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 |

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! \*\*\* |

**TUTORIAL 10: Keyboard and Screen Processing**

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

Input, output, terminate program

What are the purposes of the following functions?

1. INT 10H function 02H

Set cursor

1. INT 10H function 06H

Scroll screen

1. INT 21H function 01H

Input byte

1. INT 21H function 02H

Output byte

1. INT 21H function 07H

Input byte (no echo)

1. INT 21H function 09H

Output string

1. INT 21H function 0AH

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.

MOV AH, 02H

MOV BH, 00 ; page number

MOV DH, 11 ; row (start from 0) ; MOV DX, 0B17H

MOV DL, 23 ; column (start from 0)

INT 10H

Explain the effect of the following program segment?

MOV AX , 0600H ; Note: can simplify to AH, 06H (scroll & clear screen)

MOV BH , 71H ; (BG)(FG)H

MOV CX , 0000H ; START, Divide into 2; (ROW)|(COL)H

MOV DX , 184FH ; END, Divide into 2; (ROW)|(COL)H

INT 10H

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

1. Why the delimiter is needed?

To indicate the end of string

1. What will happen if the delimiter is missing?

The remaining data will be displayed.

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

1. What will the first byte store?

The maximum number of bytes

1. What will the second byte store?

The total number of bytes currently stored

1. What will the continuous bytes store?

The actual string

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

AH/AX

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

DL

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

MOV AH, 02H

MOV DL, 13

INT 21H

MOV DL, 10

INT 21H

For INT21H function 09H, when LEA command is applied, where will the assembler load the effective (offset) address to?

DX

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)

MOV AH, 02H

MOV BH, 00

MOV DX, 060DH

INT 21H

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 |

Write an assembly program that will prompt the user for input. The program will displays 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 |

**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)

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)
2. An I/O technique is known as Interrupt-driven I/O. explain the technique. (PYP-04/14: 5 marks)
3. Explain the FOUR (4) usages of Interrupt. (PYP-08/14: 8 marks)
4. How does Direct Memory Access (DMA) release the waiting time of CPU? (PYP-04/14: 3 marks)
5. 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)
6. Suppose you are sending a block of data from a disk to memory.
7. Identify the most appropriate I/O handling technique to support the data transfer process as mentioned. Explain your choice. (PYP-08/13: 2 marks)
8. 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)
9. Distinguish between a bus architecture and a channel architecture in terms of their characteristics and operations. (PYP-08/12: 6 marks)
10. 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)
11. 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.

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

**TUTORIAL 12: Computer Architecture**

1. Briefly explain how a system performance is affected.
2. What is the main different between loosely-coupled system and parallel system?

(PYP-01/10: 6 marks)

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)

1. What will happen when there is a cache miss? (PYP-08/13: 2 marks)
2. Analyze the impact of cache miss on the execution of instruction. (PYP-08/13: 4 marks)
3. Explain what will happened when a cache memory is full. (PYP-08/15: 4 marks)
4. List and explain TWO (2) configurations of a multiprocessing system. (PYP-08/13: 6 marks)
5. Explain the following techniques in enhancing memory performance.
6. Wider path for memory access
7. Cache memory (PYP-04/13: 4 marks)
8. What performance improvement is offered by the use of memory interleaving?
9. Explain how pipelining serves to reduce the average number of steps in the execution part of the fetch-execute cycle.
10. 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?
11. Differentiate between logical address and physical address.
12. Complete the diagram below.

