Detailed Report on the Assembly Exercise

Introduction to Assembly Language and Numeric Conversion

- What is Assembly Code?

Assembly language is a form of representation of machine code that is more easily readable by humans. It is closely tied to the architecture of the processor it is written for—in this case, the x86 architecture. Assembly provides direct control over the hardware, allowing developers to optimize the code for specific performance and operational requirements.

Assembly instructions are made up of mnemonics, such as MOV, ADD, and SUB, which represent basic operations executed by the CPU.

Hexadecimal to Decimal Conversion

Hexadecimal to Decimal Conversion
The hexadecimal system (base 16) is often used in programming as a more compact representation of binary numbers, which are the foundation of how computers operate. To convert a hexadecimal number to decimal:

- Each hexadecimal digit is multiplied by 16^n, where n is the position of the digit from the right (starting at 0).
- The sum of these products gives the equivalent decimal value.

Conversion Examples:

- 0x20 in hexadecimal = $2\times16^1 + 0\times16^0 = 32$ in decimal.
- 0x38 in hexadecimal = $3\times16^1 + 8\times16^0 = 56$ in decimal.

Decimal to Hexadecimal Conversion

Decimal to Hexadecimal Conversion

The hexadecimal system (base 16) is often used in programming as a more compact representation of binary numbers, which are the foundation of computer operation. To convert a decimal number to hexadecimal, follow these steps:

- Divide by 16: Divide the decimal number by 16. The resulting quotient will be used for the next division, while the remainder represents the least significant hexadecimal digit.
- Repeat the Process: Continue dividing by 16 using the quotient from the previous step, until the quotient becomes zero.
- The remainder at each step, after subtracting the product of the quotient and 16 from the original number, gives you the hexadecimal digits.

Practical Example:

To convert the decimal number 56 into hexadecimal:

- 56 ÷ 16 = 3.5. In this case, we only consider the integer part of the quotient. So, the quotient is 3.
- Then we calculate the remainder: 56 − (3 × 16) = 56 − 48 = 8.
- After obtaining the quotient 3 and the remainder 8:
  - The least significant digit is 8 (obtained from the remainder).
  - The next digit is 3 (obtained from the final quotient).
- Therefore, the decimal number 56 is converted into hexadecimal as 0x38.

Assembly Code Trace Resolution

01

0x00001141 <+8>: mov EAX, 0x20

- Instruction: mov EAX, 0x20
- Action: Loads the hexadecimal value 0x20 (decimal 32) into the EAX register.
- Purpose: Initializes EAX with a specific value, likely to be used in subsequent calculations or passed as a function parameter.

02

0x00001148 <+15>: mov EDX, 0x38

- Instruction: mov EDX, 0x38
- Action: Loads the hexadecimal value 0x38 (decimal 56) into the EDX register.
- Purpose: Prepares EDX with a specific value, useful for further calculations or as part of a more complex procedure.

03

0x00001155 <+28>: add EAX, EDX

- Instruction: add EAX, EDX
- Action: Adds the value in EDX to EAX, updating EAX with the result.
- Purpose: Performs an arithmetic operation by summing two values and storing the result for future use.

04

0x00001157 <+30>: mov EBP, EAX

- Instruction: mov EBP, EAX
- Action: Transfers the value in EAX (now 88) to EBP.
- Purpose: This may be used to configure EBP as a base pointer for memory access or as part of a stack frame setup.

05

0x0000115a <+33>: cmp EBP, 0xa

- Instruction: cmp EBP, 0xa
- Action: Compares EBP with 0xa (decimal 10), typically as a status or condition check.
- Purpose: Determines the program flow based on this comparison.

06

0x0000115e <+37>: jge 0x1176 <main+61>

- Instruction: jge 0x1176
- Action: Performs a jump to the specified address if EBP is greater than or equal to 10.
- Purpose: Controls the program flow based on the result of the previous comparison, branching to different code depending on the condition.

07

0x0000116a <+49>: mov eax, 0x0

- Instruction: mov EAX, 0x0
- Action: Sets EAX to zero.
- Purpose: Typically used to reset EAX in preparation for a new operation or to set a return value from a function.

08

0x0000116f <+54>: call 0x1030 <printf@plt>

- Instruction: call 0x1030
- Action: Calls the printf function located at address 0x1030.
- Purpose: Outputs a message or handles string/formatting operations, using printf to communicate with the user or for other output-related purposes.