Experiments no:02

Title: Display **s**tring

Problem Statement: Write an X86/64 ALP to accept a string and to display its length.

Objective: To learn and understand the operation of accept and display string.

Outcomes: On completion of this practical ,students will be able to
C218.1: Understand and apply various addressing modes and instruction set to implement assembly language programs

Hardware Requirement: NA

Software Requirement: Ubuntu ,NASM etc.

Theory Contents :

The 80x86 String Instructions: All members of the 80 x 86 families support five different string instructions: MOVS, CMPS, SCAS, LODS and STOS. They are the string primitives since you can build most other string operations from these five instructions.

How the String Instructions Operate: The string instructions operate on blocks (contiguous linear arrays) of memory. For example, the MOVS instruction moves a sequence of bytes from one memory location to another. The CMPS instruction compares two blocks of memory. The

SCAS instruction scans a block of memory for a particular value. These string instructions often require three operands a destination block address a source block address and (optionally) an element count. For example, when using the MOVS instruction to copy a string you need a source address a destination address and a count (the number of string elements to move).

Unlike other instructions which operate on memory the string instructions are single-byte instructions which don't have any explicit operands. The operands for the string instructions include

- The SI (source index) register
- The DI (destination index) register
- The CX (count) register
- The AX register and
- The direction flag in t e FLAGS register.

For example one variant of the MOVS (move string) instruction copies a string from the source address specified by DS:SI to the destination address sspecified by ES:DI of length CX. Likewise, the CMPS instruction compares the string pointed at by DS:SI of length CX to the string pointed at by ES: DI.

Not all instructions have source and destination operands (only MOVS and CMPS support them). For example, the SCAS instruction (scan a string) compares the value in the accumulator to values in memory. Despite their differences the 80x86's string instructions all have one thing in common - using them requires that you deal with two segments the data segment and the extra segment.

The REP/REPE/REPZ and REPNZ/REPNE Prefixes

The string instructions by themselves do not operate on strings of data. The MOVS instruction for example will move a single byte word or double word. When executed by itself the MOVS instruction ignores the value in the CX register. The repeat prefixes tell the 80x86 to do a multi-byte string operation. The syntax for the repeat prefix is:

Field: Label repeat mnemonic operand; comment

For MOVS:

REP MOVS

For CMPS:

REPE CMPS REPZ

REPNE REPNZ CMPS

For SCAS:

REPE SCAS

REPZ SCAS

REPNE SCAS

REPNZ SCAS

For STOS:

REP STOS {operands}

You don't normally use the repeat prefixes with the LODS instruction.

As you can see the presence of the repeat prefixes introduces a new field in the source line - the repeat prefix field. This field appears only on source lines containing string instructions. In your source file:

The label field should always begin in column one

- The repeat field should begin at the first tab stop and
- The mnemonic field should begin at the second tab stop.

When specifying the repeat prefix before a string instruction the string instruction repeats CX times. Without the repeat prefix the instruction operates only on a single byte word or double word.

You can use repeat prefixes to process entire strings with a single instruction. You can use the string instructions without the repeat prefix as string primitive operations to synthesize more powerful string operations.

The operand field is optional. If present MASM simply uses it to determine the size of the string to

operate on. If the operand field is the name of a byte variable the string instruction operates on bytes. If the operand is a word address the instruction operates on words. Likewise for double words. If the operand field is not present you must append a "B" "W" or "D" to the end of the string instruction to denote the size e.g. MOVSB MOVSW or MOVSD.

The Direction Flag

Besides the SI, DI and ax registers one other register controls the 80x86's string instructions - the flags register. Specifically, the direction flag in the flags register controls how the CPU processes strings.

If the direction flag is clear the CPU increments SI and DI after operating upon each string element. For example if the direction flag is clear then executing MOVS will move the byte word or double word at DS:SI to ES:DI and will increment SI and DI by one two or four. When specifying the REP prefix before this instruction the CPU increments SI and DI for each element in the string. At completion the SI and DI registers will be pointing at the first item beyond the string.

If the direction flag is set, then the 80x86 decrements si and di after processing each string element. After a repeated string operation, the si and di registers will be pointing at the first byte or word before the strings if the direction flag was set.

The direction flag may be set or cleared using the cld (clear direction flag) and std (set direction flag) instructions. When using these instructions inside a procedure keep in mind that they modify the machine state. Therefore you may need to save the direction flag during the execution of that procedure.

Assignment Questions (write-ups)

1. What is the use of addressing mode? And Explain its type.
2. Explain File Descriptor of Assembly program?
3. Explain with example string instructions of 80386 Microprocessor.
4. What is the maximum size of the instruction in 80386?
5. Which are string instructions?
6. In string operations which is by default a string source pointer?
7. In string operations which is by default a string destination pointer?
8. What is LEA? What is its use in our program?
9. Write Code of Hex to Ascii Procedure ?

Output : The output shows us actual length of string without using inbuilt string Function.

Conclusion: In this way we studied about accepting and displaying strings using 80386 microprocessors.

MPL Practical Oral Question Bank

Sr No	BL	Questions	Oral 1	Oral 2 (improvement)	Remark
1	1	What is a MacroProcessor?			
2	1	What are the types of registers and their use?			
3	2	What is addressing mode and its type?			
4	1	Which file descriptors are used in an Assembly program?			
5	2	What is a System call?Give examples of the same.			
6	1	What is the use of Macro?			
7	2	Which Assembler directive is used for macro?			

difference between microprocessor and micro controller
-->
In simple language, the main difference between a microprocessor and a microcontroller lies in their functionality and purpose:

Sign of Student

Microprocessor:
A microprocessor is a central processing unit (CPU) that is designed to execute instructions and perform computations. It is the brain of a computer system and is responsible for processing data and controlling the overall operation of the system. Microprocessors are typically used in devices that require high computational power and have more complex tasks to perform. They are commonly found in desktop computers, laptops, servers, and other computing devices.

Microcontroller:
A microcontroller is a compact integrated circuit that combines a microprocessor with other essential components like memory, input/output ports, and peripherals on a single chip. It is specifically designed for embedded systems and is often used in devices that require control, monitoring, and interaction with the external world. Microcontrollers are commonly found in appliances, consumer electronics, automotive systems, industrial machinery, and various other electronic devices. They are optimized for low-power operation and have built-in features for interacting with sensors, actuators, and other external components.

In summary, a microprocessor is a standalone CPU that focuses on processing instructions and performing computations, while a microcontroller is a complete computing system on a chip, designed for embedded systems, with additional components for control, input/output, and interfacing with the external environment.