String Instructions

Department of Computer Science & Engineering,
National Institute of Technology Calicut.

January 19, 2023

Arrays

- An Array is a continuous storage block in memory.
- Each element of the array have the same size.
- We access each element of the array using:
  - ⋆ Base address/address of the first element of the array.
  - ⋆ Size of each element of the array.
  - ⋆ Index of the element we want to access.
- In NASM there is no array element accessing/dereferencing operator like [ ] in C / C++ / Java .
- We compute the address of each element using an iterative control structure and traverse though the elements of the array.

Strings

- **Strings are stored in memory as array of characters.**
  - **Declaring/ Initializing a string**
    section .bss

    string: resb 50
  - **Reading a string**
    Pseudo Code:
    i=0
    while(num!='\n')
    read(num)
    *(arr+i)=num
    i++
    endwhile

Strings

- NASM Code for reading a string

```
read_array:
pusha

reading:
push ebx
mov eax, 3
mov ebx, 0
mov ecx, temp
mov edx, 1
int 80h

pop ebx
cmp byte[temp], 10 ;; check if the input
is 'Enter'
je end_reading
inc byte[string_len]
mov al,byte[temp]
mov byte[ebx], al
inc ebx
jmp reading

end_reading:
mov byte[ebx], 0 ;; Similar to putting a
null character at the end of a string
mov ebx, string
popa
ret
```

Strings

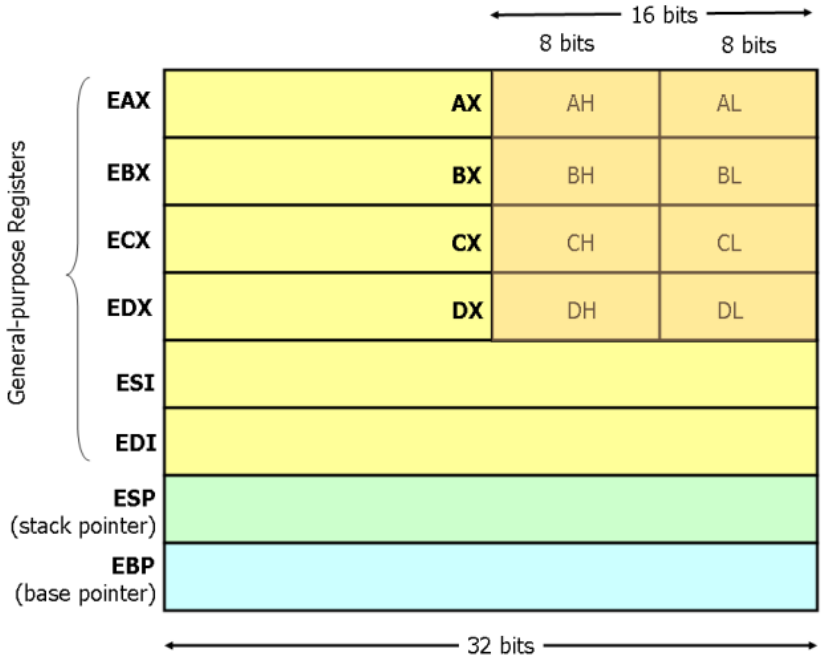
- NASM Code for printing a string

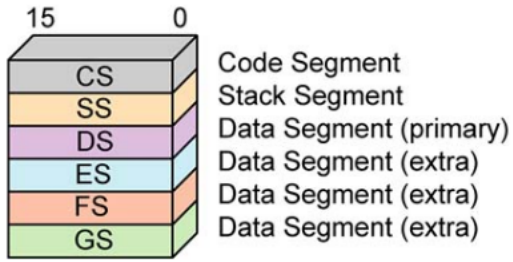
```
print_array:
pusha
mov ebx, string

printing:
mov al, byte[ebx]
mov temp, al
cmp byte[temp], 10
je end_printing

push ebx
mov eax, 4
mov ebx, 1
mov ecx, temp
mov edx, 1
int 80h
pop ebx
inc ebx
jmp printing

end_printing :
popa
ret
```

String Operations

- x86 Processors have a set of instructions designed specially to do string operations called String Instructions
  - They use index registers(ESI EDI) and increments/decrements either one or both the registers after each operation.
  - Depending on the value of Direction Flag(DF) it either increments or decrements for the DI and SI registers during string operations
  - The following instructions are used to set the value of DF manually:
    - i) CLD - Clears the Direction Flag (D=0)
    - ii) STD - Sets the Direction Flag (D=1)
  - NB: Always make sure to set the value of Direction Flags explicitly, else it may lead to unexpected errors.
  - Make sure to have DS to be the segment base of Source string and ES to be the segment base of Destination String.

Registers


16 bits
8 bits
8 bits
General-purpose Registers
EAX
AX
AH
AL
EBX
BX
BH
BL
ECX
CX
CH
CL
EDX
DX
DH
DL
ESI
EDI
ESP
(stack pointer)
EBP
(base pointer)
32 bits


15
0
CS
SS
DS
ES
FS
GS
Code Segment
Stack Segment
Data Segment (primary)
Data Segment (extra)
Data Segment (extra)
Data Segment (extra)

String Instructions

Each string instruction allows data transfers that are either a single byte, word, or double word.

1. Reading an array element to reg(AL/AX/EAX)

- To copy one element from an array to the register
- LODSx: x = B / W / D - Load String Instruction
- LODSB
  AL = byte[DS:ESI]
  ESI = ESI ± 1
- LODSW
  AX = word[DS : ESI]
  ESI = ESI± 2
- LODSD
  EAX = dword[DS : ESI]
  ESI = ESI± 4

String Instructions

2. Storing a reg(AL/AX/EAX) to an array

- To copy one element from a register to an array.
- STOSx: x = B / W / D - Store String Instruction
- STOSB
  byte[ES:EDI] = AL
  EDI = EDI ±; 1
- STOSW
  word[ES : EDI] = AX
  EDI = EDI± 2
- STOSD
  dword[ES : EDI] = EAX
  EDI = EDI±; 4

String Instructions

Eg: Program to increment the value of all array elements by 1

```
section .data
array1: db 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
section .text
global _start
start:
CLD ;Clears the Direction Flag
mov esi, array1 ;Copy Base address of array to index registers
mov edi, array1
mov ecx, 10 ;No: of element in the array
increment:
LODSB
INC al
STOSB
loop increment
```

.....

String Instructions

3. Memory Move Instructions

- To copy the elements of one array/string to another.
- MOVSx: x = B / W / D - Move String Instruction
- MOVSB
  byte[ES:EDI] = byte[DS:ESI]
  ESI = ESI ±; 1
  EDI = EDI±; 1

String Instructions

Eg:Program to copy elements of an array to another

```
section .data
array1: dd 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
section .bss
array2: resd 10
section .text
global_ start
_start:
CLD ;Clears the Direction Flag
mov esi, array1 ;Copy Base address of array to index registers
mov edi, array2
mov ecx, 10 ;No: of element in the array
copy:
MOVSD
loop copy
.....
.....
```

String Instructions

4.REP - Repeat String Instruction

- Repeats a string instruction.
- Number of times repeated is equal to the value of ecx register.
- **Eg: Previous program using REP instruction.**

```
section .data
array1: dd 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
section .bss
array2: resd 10
section .text
global _start
start:
CLD ;Clears the Direction Flag
mov esi, array1 ;Copy Base address of array to index registers
mov edi, array2
mov ecx, 10 ;No: of element in the array
REP MOVSD
```

String Instructions

5.Compare Instructions CMPSx : x = B / W / D - Compares two array elements and affects the CPU Flags

6.Scan Instructions

- SCASx : x = B / W / D - Compares a register(AL/AX/EAX) with an array element.
- Affects the CPU Flags.
- SCASB
  Compares value of AL with byte[ES:EDI]
  EDI = EDI ± 1

String Instructions

Eg: Scanning an array for an element

```
section.data
array1 : db 1, 5, 8, 12, 13, 15, 28, 19, 9, 11
section .text
global _start
start :
CLD; Clears the Direction F lag
move di, array1 ; Copy Base address of array to index registers
move cx, 10 ; N o : of element in the array
mov al, 15 ; Value to be searched
scan :
SCASB
je found
loop scan
jmpnotfound
.....
.....
```

Conclusion

- Five basic instructions for processing strings : 1. MOVS 2. LODS 3. STOS 4. CMPS 5. SCAS

THANK YOU

References

[1] The Intel Microprocessors : Architecture, Programming, and Interfacing, Barry B. Brey, 8th Ed., Prentice Hall, 2009.
[2] Introduction to NASM, Jayaraj P B Saidalavi Kalady, NIT Calicut, 2019.