Page :
Date : / /

Assignment 2

Date :

Title : Write X86/64 ALP to perform Non-overlapped and overlapped block transfer (with and without string specific instructions). Block containing data can be defined in the data segment.

Objective :

i) To learn overlapped/non-overlapped data transfer in segments
ii) Block transfer instruction of 8086.
iii) Data storage in the memory and segments.

Outcome :

Students will study different block transfer instructions and also understood block transfer within different segments.

Theory :

One of the frequent operations used in programming is shifting/transferring the data from one memory location to another memory location.

The data can be transferred either in overlapped fashion or non-overlapped fashion.

1) Overlapped Fashion.

In this case the data stored on different address changes due to overlapping of the data

Page :
Date : / /

Example :

Before :

Address	Value
101	10
102	20
103	30
104	40
105	50

After

Address	Value
101	10
102	20
103	30
104	10
105	20
106	30
107	40
108	50

In the above example, data stored on memory location 104 and 105 gets overlapped with data from memory location 101 and 102.

2) Non-overlapped.

In this case, block of data is transferred from one memory location to another memory location.

Example :

Before

Address	Value
101	10
102	20
103	30
104	40
105	50

After

Address	Value
201	10
202	20
203	30
204	40
205	50

The string instructions use register rax, rsi, rdi for special purposes. Registers rax or its sub-registers eax, ax, al are used to hold a specific value. Register rsi is the source index register and rdi is the destination index. None of the string instructions need operands.

The string operations update the source and/or destination registers after each use. This updating is managed by the direction flag (DF). If the the DF is 0 then the registers are increased by the size of the data item after each use. If DF is 1 then the register are decreased after each use.

movsb:

This instructions moves bytes from the address specified by rsi to the address specified by rdi. The other movs instructions like movsq, movsw movsd move 2, 4 or 8 bytes of data from [rdi] to [rsi]. The data moved is not stored in a register and no flags are affected. After each data item is moved, the rdi and rsi registers are advanced 1, 2, 4 or 8 bytes depending on the size of the data item.

Set/clear direction

The clear direction cld instruction clears the direction flag to 0, which means to process increasing addresses with the string operations. The set direction (std) sets the direction flag to 1.

Page :
Date : / /

Repeat instructions.

1) REP : This instruction repeats a string instruction the number of times specified in the count register (rcx).
2) REPE : This instruction repeats a string instruction the number of times specified in the counter register (rcx) while zero flag (ZF) is set.
3) REPNE : This instruction repeats a string instruction the number of times specified in the count register (rcx) while ZF != 0.

Algorithm :

i) Printing array:

1. Set counter to number of elements in array.
2. Initialize rsi with array.
3. Display the content of rsi.
4. Increment rsi
   Decrement counter.
5. If counter != 0 goto step 3
6. end

ii) Block transfer.

a) Non-overlapping and without string.

1) Start.
2) Initialize rsi register with source address and rdi register with destination array.
3) Initialize counter to size of array.
4) move content of rsi regesiter into a temporary data register and the move it to the address

specified by rdi.
5) Increment rsi and rdi
6) Decrement counter.
7) If counter != 0 goto step4
8) Stop.

b) Non-overlapping with string.

1) Start.
2) Initialize rsi and rdi with source address and destination address respectively.
3) Initialize rcx register with the number of elements in array.
4) Clear direction flag if set using cld.
5) Use movsb (if data is in byte) instruction to transfer data from source to destination.
6) Repeat the same instruction using rep instruction while [rcx] != 0.
7) End.

c) Overlapping without string instruction.
1) Start
2) Initialize rsi with address of array and set counter to number of elements in array.
3) Copy the contents of array to another array.
4) After copying initialize rsi with address arr+2 (if user wants to start overlapping from third address) and rdi with address of array and rcx = size of array.
5) Move [rdi] to rdx and move contents of rdx to [rsi] (since memory to memory transfer cannot

take place)
6) Increment rsi and rdi
7) Decrement counter.
8) If counter != 0 goto step 5.
9) end.

d) Overlapping with string instructions.
1) Start.
2) Initialize rsi with address of array and set counter to number of elements in array.
3) Copy the contents of array to another array using string instruction.
4) After copying, initialize rdi with address arr+2 and rsi with address of array1 and set counter register rcx to size of array1.
5) Use movsb (if data in byte) string instructio to transfer data.
6) Repeat the instruction till counter != 0
7) end.

Page
Date / /

Testcases:

1) Non overlapping

Testcase	Output	Expected	Success/Fail.
arr: db 10,11,12 13,14,15	arr: 10,11,12 13,14,15,10, 11,12,13,14,15	Same as output	Success
arr1: db 01,10, 21,23,98	arr1: 01,10,21 23,98,01,10,21 23,98	Same as output	Success.

2) Overlapping

Testcase	Output	Expected	Success/Fail.
arr: db 10,11,12 13,14,15	arr: 10,11, 10,11,12,13, 14,15	Same as output	Success
arr1: db 01,10, 21,23,98	arr1: 01,10, 21,23,01,10 21,23,98	Same as output	Success.

Conclusion: We learnt how to perform overlap and non-overlap block transfer with and without using string instructions.