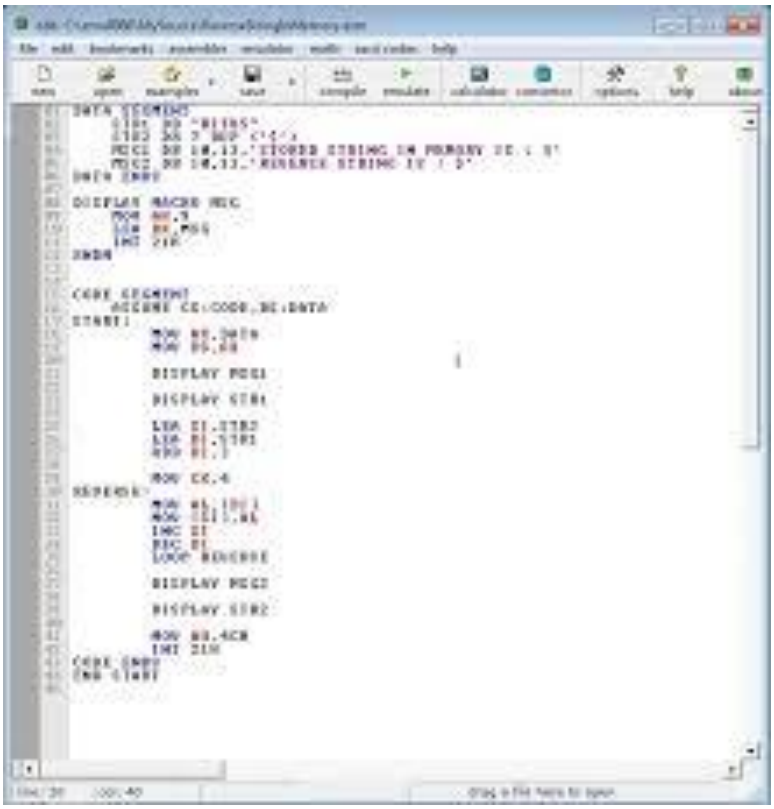


Writing a Binary File in Assembly

ARRAY OF DOUBLE WORDS

MICROPROCESSOR AND EMBEDDED SYSTEM

11.06.2020

170316064 Fatma KURTULUŞ

180316054 Yağızhan Arslan AKINCI

170316063 Emrehan ÖZYÜREK

Problem

In our problem we are going to write an array of double words into a binary file. An array of double words contains elements where each element takes 32 bit of memory. An Example declaration of array of double words in assembly is like as:

```
Array dd 1234h, 5454h, 2324h
```

Here we must use the word dd. This dd means define double word.

In our program we have to write an array of double words into a text file as binary. To do so we must convert each element of the array in binary. This binary conversion can be done very easily. Using test instruction. The test instruction test a register to be equal to something.

Our Program Flow:

Our program flow is very simple. We have the following steps:

1. At the very first the program we open the file where the data should be written
2. The handle of the file is written in the variable “handle”
3. We call a function called “writeArray”
4. In the write array function:
    a. We take the register si as the index of the array
    b. Each element of the array is traversed one by one
    c. For each element we convert its integer value into binary string and write to the file at same time
    d. After completing write the function returns.
5. Finally file is closed
6. Program also closed.

Code Blocks with Description

Our system works as described in the rules below:

1. the emulator emulates all drive paths in c:\emu8086\vdrive\
   for example: the real path for "c:\test1" is
   "c:\emu8086\vdrive\c\test1"
2. paths without drive letter are emulated to c:\emu8086\MyBuild\
   for example: the real path for "myfile.txt" is
   "c:\emu8086\MyBuild\myfile.txt"
3. 3. if compiled file is running outside of the emulator rules 1 and 2 do
   not apply.

```
org  100h
jmp start
nl db 0AH,0DH
nl_size = $ - offset nl
file1 db "c:\test1\file1.txt", 0
handle dw ?

zero db "0"
one  db "1"
array dd 1234h, 5454h,2324h
array_size =  $ - offset array

start:
mov ax, cs
mov dx, ax
mov es, ax
```

In this section we write where to save our file1.txt file, then we save 1 and 0 to our database and finally define our array. The hexadecimal numbers defined in the array will be converted to decimals.

```
mov ah, 3ch
mov cx, 0
mov dx, offset file1
int 21h

mov handle, ax
```

We create and open file: c:\emu8086\vdrive\C\test1\file1.txt and handle contains the file handler or descriptor

```
call     writeArray


mov ah, 3eh
mov bx, handle
int 21h
err2:
nop

ret
```

In the first line, we write our array to our file and in the rest of the code we close our file.

```
writeArray:
    push    si

    mov     si, offset Array
    mov     cx,  Array_Size - 1



ShowIt:
    mov     bl, [si]

    push cx

    mov cx,8
    write:

    test bl, 10000000b
    jz szero
```

```
push ax
push bx
push cx
push dx
mov ah, 40h
mov bx, handle
mov dx, offset one
mov cx, 1
int 21h
pop dx
pop cx
pop bx
pop ax

jmp write_end
szero:

push ax
push bx
push cx
push dx
mov ah, 40h
mov bx, handle
mov dx, offset zero
mov cx, 1
int 21h
pop dx
pop cx
pop bx
pop ax
```

Brings byte to be used from memory. It checks whether the brought byte is 0 or 1.

In the first part of the code, it writes 0s in our array. In the rest of the code, it writes the 1s in our array.

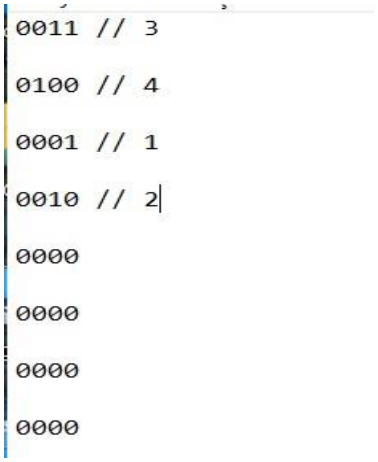
```
write_end:
    shl bl, 1
    cmp cx,5
    jne loop_end

    push ax
     push bx
     push cx
     push dx
     mov ah, 40h
     mov bx, handle
     mov dx, offset nl
     mov cx, nl_size
     int 21h
     pop dx
     pop cx
     pop bx
     pop ax

    loop_end:
    loop write
```

Here we can see the output of the code, so after writing 4 bytes it goes to the bottom line. 4 bytes, correspond to a digit in our hexadecimal number. So if the number in our hexadecimal number is 1 (4 bytes), our binary number will be 0001 consisting of four digits.

Output



Endian style coding is used in our system. Endian style; Set up the numbers in the array as 4 bytes and write the last 2 numbers first, then the first two numbers, and the rest is filled with 0s. For example; If our number is 1234, our output will be 3412.

You can see this in our output.

Conclusion:

Finally the program is a very important program in the sense that it can write an array of double words into a file as binary string.