**Computer Organization & Assembly Language**

**Lab 04**

**Topics:**

1. Directives ($, =, equ, textequ)
2. Current Location Counter
3. Signed DataTypes
4. **Symbolic Definitions:**

A symbolic definition is created by associating an identifier (a symbol) with an integer expression or some text. Symbols do not reserve any memory storage, and their values cannot be changed at runtime. Different directives will be used to create different symbols.

‘$’ is the current location counter and it tells the current location of the variable. It returns the offset associated with the current program statement.

*list db 10,20,30,40,50*

*listlen = ($-list)*

‘Equal sign, =’ directive associates a symbol with an integer expression.

*Name = expression # count = 100*

Any symbol defined with the equal sign directive can be redefined in the code later.

‘equ’ directive associates a symbolic name with an integer expression or some arbitrary text.

*name equ expression # a equ 2+3*

*name equ symbol # a equ var (var defined by = or equ)*

*name equ <text> # a equ <2+3>*

Any symbol defined with the equ directive cannot be redefined in the same source code file.

‘textequ’ directive creates a text macro. There are three different formats:

*var1 = 5*

*name textequ text # count textequ %(var1 \* 2)*

*name textequ <text macro> # move textequ <mov>*

*name textequ %constant\_expression # Copy\_AL textequ <move al,count>*

Any symbol defined with the directive textequ can be redefined at any time.

1. **Current Location Counter:**

One of the most important symbols of all, shown as $, is called the current location counter.

It tells the current location of the variable and returns the offset associated with the current program statement.

my\_string db “Hello World”

my\_string\_size = ( $ - my\_string )

1. **Signed DataTypes:**

signed data types refer to variables or values that can represent both positive and negative numbers. Unlike unsigned data types, which only store positive numbers, signed data types use one bit (usually the most significant bit, or MSB) to indicate the sign of the number

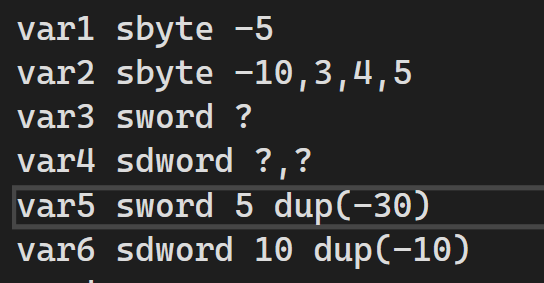
*var sbyte 5*

*var1 sword 10*

*var3 sdword 20*

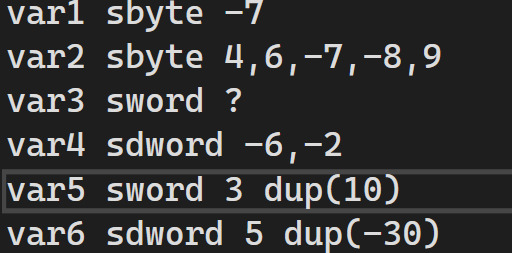
**Tasks:**

1. Using the array declared and initialized in above question, you need to add any number at the even/odd indices of the array. (even/odd determined based on your roll number.)
2. Using the array in question 2, replace an element in array at the index based on the last digit of your roll number by a variable. View the new array in memory.



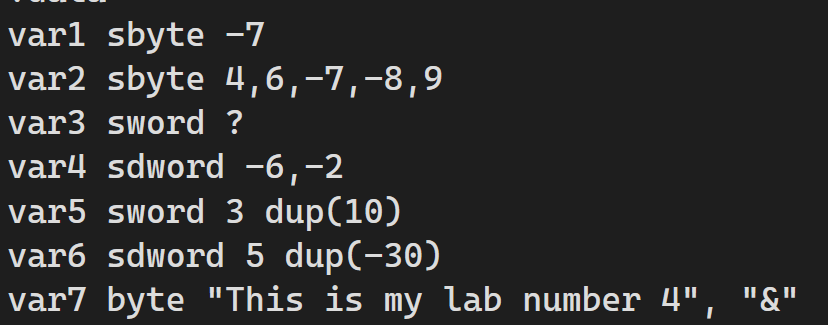
Fill the memory after execution and also write which format the it is saved in your computer(little endian or big endian).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F |
| 0000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0010 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0020 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0030 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1. 

Fill the memory after execution and also write which format the it is saved in your computer(little endian or big endian).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F |
| 0000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0010 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0020 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0030 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |



1. Fill the memory after execution and also write which format the it is saved in your computer(little endian or big endian).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F |
| 0000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0010 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0020 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0030 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |