NEW YORK CITY COLLEGE OF TECHNOLOGY

THE CITY UNIVERSITY OF NEW YORK

Department of Computer Engineering Technology

300 Jay Street, Brooklyn, NY 11201-1909

LAB REPORT

CET 3510 – OL71

(MICROCOMPUTER SYSTEMS TECHNOLOGY LABORATORY)

LAB #5

Addressing Modes

Name: Puja Roy

Date: 10/10/21

Due Date: 10/17/21

Table of Contents

Objective:

The objective of this lab is to be familiar with the operation of each data-addressing mode. It is required to utilize and select the appropriate addressing mode for a give task in the program. This experiment will allow to exam a variety of different data-addressing modes including register addressing such as MOV, AX, CX, intermediate addressing consisting of MOV AX, Aa, direct addressing, register indirect addressing and register relative addressing.

Materials:

- Microsoft Visual Studio C++ Community Edition 2019

Procedure:

1. First, open Microsoft Visual Studio C++ Community Edition 2019
2. Then, type program#1, compile and run the program.
3. Then modify example 2 to write program#2.
4. Draw a table of the variable value and the memory address of the variable.
5. Lastly, analyze the output.

Program#1

```
#include <iostream>
using namespace std;

int main()
{
	const int n = 5; // the dimension of the array
	unsigned short u16_arr[n] = { 0x0041, 0x0052, 0x1169, '12', 'aA' };
	unsigned short* uPtr;
	unsigned short u0, u1, u2, u3, u4;
	int i;  // used as index
	int u0_addr, u1_addr, u2_addr, u3_addr, u4_addr;

	//Display the hexadecimal and decimal values for each element of the unsigned
short array
	cout <<
"++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n";
	cout << "--------------The value of each element of the 16-bit array--------\n";
	for (i = 0; i < n;i++)
	{
		cout << "The value and the size of element " << dec << i;
		cout << "in an array are 0x" << hex << u16_arr[i]
			<< "and" << sizeof(u16_arr[i]) << "byte(s)" << endl;
	}
	printf("-----------------------------------------------------------\n");
	uPtr = u16_arr;

	//Display the 32-bit address in hexadecimal format
	cout << "The memory address of the array is 0x" << hex << uPtr << endl;
```

```
	_asm
	{
		//EBX holds the address of element 0 in the array
		mov EBX, uPtr;

		/*u0 holds the retrieved value from the indirect address contained in
register EBX*/
		mov AX, [EBX];
		mov u0, AX;
		mov u0_addr, EBX;

		/*EBX is increased by 2 due u16_arr is an unsigned short array (2 bytes)
EBX holds the address of element 1 in the array*/
		mov ECX, 2H;
		add EBX, ECX;

		/*u1 holds the retrieved value from the indirect address contained in
register EBX*/
		mov AX, [EBX];
		mov u1, AX;
		mov u1_addr, EBX;

		/*EBX+2H holds the address of element 2 in the array u2 holds the retrieved
value from the register relative addressing*/
		mov AX, [EBX + 2H];
		mov u2, AX;

		//EDI holds the address of the element in an array
		mov EDI, EBX;
		add EDI, 2H;
		mov u2_addr, EDI;

		/*EBX+4H holds the address of element 3 in the array u3 holds the retrieved
value from the register relative addressing*/
		mov AX, [EBX + 4H];
		mov u3, AX;

		// EDI holds the address of element 3 in the array
		mov EDI, EBX;
		add EDI, 4H;
		mov u3_addr, EDI;

		/*EBX+6H holds the address of element 4 in the array u4 holds the retrieved
value from the register relative addressing*/
		mov AX, [EBX + 6H];
		mov u4, AX;

		//EDI holds the address of element 4 in the array
		mov EDI, EBX;
		add EDI, 6H;
		mov u4_addr, EDI;
	}

	cout << "-------------------------------------------------------" << endl;
	cout << "The retrieved values 0x" << hex << u0 << "\t at the address of 0x" << hex
<< u0_addr << endl;
	cout << "The retrieved values 0x" << hex << u1 << "\t at the address of 0x" << hex
<< u0_addr << endl;
```

```
	cout << "The retrieved values 0x" << hex << u2 << "\t at the address of 0x" << hex
<< u0_addr << endl;
	cout << "The retrieved values 0x" << hex << u3 << "\t at the address of 0x" << hex
<< u0_addr << endl;
	cout << "The retrieved values 0x" << hex << u4 << "\t at the address of 0x" << hex
<< u0_addr << endl;

	system("pause");
	exit(0);
	return 0;
}
```

Output:

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
-------------The value of each element of the 16-bit array--------
The value and the size of element 0in an array are 0x41and2byte(s)
The value and the size of element 1in an array are 0x52and2byte(s)
The value and the size of element 2in an array are 0x1169and2byte(s)
The value and the size of element 3in an array are 0x3132and2byte(s)
The value and the size of element 4in an array are 0x6141and2byte(s)
--------------------------------------------------------
The memory address of the array is 0x0133FD90
--------------------------------------------------------
The retrieved values 0x41        at the address of 0x133fd90
The retrieved values 0x52        at the address of 0x133fd90
The retrieved values 0x1169      at the address of 0x133fd90
The retrieved values 0x3132      at the address of 0x133fd90
The retrieved values 0x6141      at the address of 0x133fd90
Press any key to continue . . .
```

Program#2

```
#include <iostream>
using namespace std;

int main()
{
	const int n = 5; // the dimension of the array
	unsigned short u16_arr[n] = { 0x0041, 0x0052, 0x1169, '12', 'aA' };

	int sign32Arr[n] = { 0xFFFFBBBB, 0XCBCBCBCB, 0X11111111, 0X12345678, 0X000002FA };

	unsigned short* uPtr;
	unsigned short u0, u1, u2, u3, u4;

	int* signedPtr;
	int s0, s1, s2, s3, s4;

	int i;  // used as index
	int u0_addr, u1_addr, u2_addr, u3_addr, u4_addr;
```

```
	//Display the hexadecimal and decimal values for each element of the unsigned
short array
	cout <<
"++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n";
	cout << "--------------The value of each element of the 16-bit array--------\n";
	for (i = 0; i < n;i++)
	{
		cout << "The value and the size of element " << dec << i;
		cout << "in an array are 0x" << hex << u16_arr[i]
			<< "and" << sizeof(u16_arr[i]) << "byte(s)" << endl;
	}
	printf("------------------------------------------------------------\n");
	uPtr = u16_arr;

	//Display the 32-bit address in hexadecimal format
	cout << "The memory address of the array is 0x" << hex << uPtr << endl;
	_asm
	{
		//EBX holds the address of element 0 in the array
		mov EBX, uPtr;

		/*u0 holds the retrieved value from the indirect address contained in
register EBX*/
		mov AX, [EBX];
		mov u0, AX;
		mov u0_addr, EBX;

		/*EBX is increased by 2 due u16_arr is an unsigned short array (2 bytes)
EBX holds the address of element 1 in the array*/
		mov ECX, 2H;
		add EBX, ECX;

		/*u1 holds the retrieved value from the indirect address contained in
register EBX*/
		mov AX, [EBX];
		mov u1, AX;
		mov u1_addr, EBX;

		/*EBX+2H holds the address of element 2 in the array u2 holds the retrieved
value from the register relative addressing*/
		mov AX, [EBX + 2H];
		mov u2, AX;

		//EDI holds the address of the element in an array
		mov EDI, EBX;
		add EDI, 2H;
		mov u2_addr, EDI;

		/*EBX+4H holds the address of element 3 in the array u3 holds the retrieved
value from the register relative addressing*/
		mov AX, [EBX + 4H];
		mov u3, AX;

		// EDI holds the address of element 3 in the array
		mov EDI, EBX;
		add EDI, 4H;
		mov u3_addr, EDI;
```

```
		/*EBX+6H holds the address of element 4 in the array u4 holds the retrieved
value from the register relative addressing*/
		mov AX, [EBX + 6H];
		mov u4, AX;

		//EDI holds the address of element 4 in the array
		mov EDI, EBX;
		add EDI, 6H;
		mov u4_addr, EDI;
	}

	cout << "-------------------------------------------------------" << endl;
	cout << "The retrieved values 0x" << hex << u0 << "\t at the address of 0x" << hex
<< u0_addr << endl;
	cout << "The retrieved values 0x" << hex << u1 << "\t at the address of 0x" << hex
<< u0_addr << endl;
	cout << "The retrieved values 0x" << hex << u2 << "\t at the address of 0x" << hex
<< u0_addr << endl;
	cout << "The retrieved values 0x" << hex << u3 << "\t at the address of 0x" << hex
<< u0_addr << endl;
	cout << "The retrieved values 0x" << hex << u4 << "\t at the address of 0x" << hex
<< u0_addr << endl;


	/*****************************************************32bit Array
Section***************************************** */
	//Display the hexadecimal and decimal values for each element of the unsigned
short array
	cout <<
"++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n";
	cout << "--------------The value of each element of the 32-bit array--------\n";
	for (i = 0; i < n;i++)
	{
		cout << "The value and the size of element " << dec << i;
		cout << "in an array are 0x" << hex << sign32Arr[i]
			<< "and" << sizeof(sign32Arr[i]) << "byte(s)" << endl;
	}
	printf("-------------------------------------------------------\n");
	signedPtr = sign32Arr;

	//Display the 32-bit address in hexadecimal format
	cout << "The memory address of the array is 0x" << hex << uPtr << endl;
	_asm
	{
		//EBX holds the address of element 0 in the array
		mov EBX, signedPtr;

		/*u0 holds the retrieved value from the indirect address contained in
register EBX*/
		mov EAX, [EBX];
		mov s0, EAX;
		mov u0_addr, EBX;

		/*EBX is increased by 4 due signed32Arr is an unsigned short array (2
bytes) EBX holds the address of element 1 in the array*/
		mov ECX, 4H;
		add EBX, ECX;
```

```
		/*u1 holds the retrieved value from the indirect address contained in
register EBX*/
		mov EAX, [EBX];
		mov s1, EAX;
		mov u1_addr, EBX;

		/*EBX+4H holds the address of element 2 in the array u2 holds the retrieved
value from the register relative addressing*/
		mov EAX, [EBX + 4H];
		mov s2, EAX;

		//EDI holds the address of the element in an array
		mov EDI, EBX;
		add EDI, 4H;
		mov u2_addr, EDI;

		/*EBX+8H holds the address of element 3 in the array u3 holds the retrieved
value from the register relative addressing*/
		mov EAX, [EBX + 8H];
		mov s3, EAX;

		// EDI holds the address of element 3 in the array
		mov EDI, EBX;
		add EDI, 8H;
		mov u3_addr, EDI;

		/*EBX+6H holds the address of element 4 in the array u4 holds the retrieved
value from the register relative addressing*/
		mov EAX, [EBX + 0xC];
		mov s4, EAX;

		//EDI holds the address of element 4 in the array
		mov EDI, EBX;
		add EDI, 0xC;
		mov u4_addr, EDI;
	}

	cout << "-------------------------------------------------------" << endl;
	cout << "The retrieved values 0x" << hex << s0 << "\t at the address of 0x" << hex
<< u0_addr << endl;
	cout << "The retrieved values 0x" << hex << s1 << "\t at the address of 0x" << hex
<< u0_addr << endl;
	cout << "The retrieved values 0x" << hex << s2 << "\t at the address of 0x" << hex
<< u0_addr << endl;
	cout << "The retrieved values 0x" << hex << s3 << "\t at the address of 0x" << hex
<< u0_addr << endl;
	cout << "The retrieved values 0x" << hex << s4 << "\t at the address of 0x" << hex
<< u0_addr << endl;

	system("pause");
	exit(0);
	return 0;
}
```

Output:

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
--------------The value of each element of the 16-bit array--------
The value and the size of element 0in an array are 0x41and2byte(s)
The value and the size of element 1in an array are 0x52and2byte(s)
The value and the size of element 2in an array are 0x1169and2byte(s)
The value and the size of element 3in an array are 0x3132and2byte(s)
The value and the size of element 4in an array are 0x6141and2byte(s)
------------------------------------------------------
The memory address of the array is 0x00AFFD2C
------------------------------------------------------
The retrieved values 0x41        at the address of 0xaffd2c
The retrieved values 0x52        at the address of 0xaffd2c
The retrieved values 0x1169      at the address of 0xaffd2c
The retrieved values 0x3132      at the address of 0xaffd2c
The retrieved values 0x6141      at the address of 0xaffd2c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
--------------The value of each element of the 32-bit array--------
The value and the size of element 0in an array are 0xffffbbbband4byte(s)
The value and the size of element 1in an array are 0xcbcbcbcband4byte(s)
The value and the size of element 2in an array are 0x11111111and4byte(s)
The value and the size of element 3in an array are 0x12345678and4byte(s)
The value and the size of element 4in an array are 0x2faand4byte(s)
------------------------------------------------------
The memory address of the array is 0x00AFFD2C
------------------------------------------------------
The retrieved values 0xffffbbbb  at the address of 0xaffd10
The retrieved values 0xcbcbcbcb  at the address of 0xaffd10
The retrieved values 0x11111111  at the address of 0xaffd10
The retrieved values 0x12345678  at the address of 0xaffd10
The retrieved values 0x2fa       at the address of 0xaffd10
Press any key to continue . . .
```

Table:

Data Type	Variable Name	Memory contents	Address
Unsigned short	u16_arr[0]	0x41	0x6ffba0
Unsigned short	u16_arr[1]	0x52	0x6ffba2
Unsigned short	u16_arr[2]	0x1169	0x6ffba4
Unsigned short	u16_arr[3]	0x3132	0x6ffba6
Unsigned short	u16_arr[4]	0x6141	0x6ffba8
Unsigned int	u32_arr[0]	0x41	0x118f9b4
Unsigned int	u32_arr[1]	0x52	0x118f9b8
Unsigned int	u32_arr[2]	0xabcd1169	0x118f9bc
Unsigned int	u32_arr[3]	0x313233334	0x118f9c0
Unsigned int	u32_arr[4]	0x61414263	0x118f9c4
Signed int	i32_arr[0]	0x41	0x118f908
Signed int	i32_arr[1]	0x52	0x118f90c

Conclusion:

Throughout this lab, I was able to familiarize with the operation of each data addressing mode. I used and selected the appropriate addressing mode for an assigned task. I also examined different data addressing modes including register addressing, immediate addressing, direct addressing, register indirect addressing, and register relative addressing.