CS118 Lab2

The purpose of this program is to understand the division method for converting decimal numbers (base 10) to hexadecimal numbers (base 16). This method is implemented in C. The program in C is then used as a guideline to write the same program in assembly language. In this program, the decimal number ”21602” is converted to it’s hexadecimal form ”5462”.

All variables are made to be global to increase efficency and to make it easier to translate into assembly code.

C source code written to file lab.c

```
#include <stdlib.h>
#include <stdio.h>
char decimalNumber[] = ”21602”;
#define numberOfDecDigits (sizeof(decimalNumber)−1)
int i; //index
int decimal; //int form of number stored in char decimalNumber
int numHexDigits; //number of hexadecimal digits
char hexDigits[numberOfDecDigits];
int temp; int decDigit; int hexDigit;
char hexarray[] = ”0123456789ABCDEF”;
```

ASM source code written to file lab.s

```
.data
    decimalNumber: .string ”21602”
    .equ numberofDecDigits, (. − decimalNumber − 1);
    hexarray: .ascii ”0123456789ABCDEF”
.bss
    .lcomm i, 4
    .lcomm numHexDigits, 4
    .lcomm hexDigits, 4
    .lcomm temp, 4
    .lcomm decDigit, 4
    .lcomm hexDigit, 4
```

All arrays in the C program: decimalNumber, hexDigits, hexarray, are initialized and hence put in the .data section of the ASM program and defined as string arrays. decimalNumber contains the decimal number that must be converted. The program has not hardcoded this number. The define variable which determines the size of the decimalNumber array, excluding the null memory location (hence -1) is stored using .equ in the ASM program as a symbolic constant. Using the memory allocation locator, we are able to replicate this line of C code. The integer variables are all uninitialized and hence put in the .bss section. The final answer will be stored in the array hexDigits.

CS118 Lab2

This is the exit function in C and the exit procedure in ASM that will terminate the program.

C source code appended to file lab.c

```
void bye()
    {
    exit(0);
    }
```

ASM source code appended to file lab.s

```
bye: #procedure 1
mov $1, %eax
mov $0, %ebx
int $0x80
ret
```

- The C program contains the exit function ”exit(0);”
- The same function in ASM takes three lines. For this lab, I have created a procedure called bye which matches the name of the exiting function in C.
- The literal value 1 is stored in the eax register
- The literal value 0 is moved to the ebx register

CS118 Lab2

This is the function which will be called in the accummulating function to convert the char array storing the decimal number into an integer.

C source code appended to file lab.c

```
void getdecimalDigit()
{
 decDigit = decimalNumber[i] - '0';
 return;
}
```

ASM source code appended to file lab.s

```
getdecimalDigit: #procedure 2
    mov $decimalNumber , %edi
    mov i , %ecx
    movb (%edi,%ecx,1), %ebx
    sub $'0', %ebx
    mov %ebx,decDigit
    ret
```

Essentially, the decimal integer value is stored in decDigit by subtracting the ascii value of 0 from the ascii elements in the decimalNumber array. The dollar sign is equal to in C code. This allows us to dereference a memory location. The register edi used for string procedures. The ecx register was used because ecx used for loops commonly. The array that is located in edi, takes index from ecx. Each element is 1 byte long. () dereferences the memory location. The b in movb moves and subtracts one byte.

CS118 Lab2

This is the function which will be called in the while loop which converts the decimal integer to a hexadecimal value.

C source code appended to file lab.c

```
void gethexDigit()
{
    hexDigit = decimal % 16;  //gives position
    decimal /= 16;
    return;
}
```

ASM source code appended to file lab.s

```
gethexDigit: #procedure 3
    mov decimal, %eax
    mov
    idivl $16
    mov %edx, hexDigit
    mov %eax, decimal
    ret
```

The gethexDigit function and procedure is used to calculate and retrive the hexadecimal value positions and store them in hexDigit. hexDigit is later used as an index for the hexarray to retrive the corresponding hexadecimal values. In the ASM code, the register eax holds the quotient, in this case decimal. idivl corresponds to the decimal /= 16 line in C. As per the division algorithm, we divide the

CS118 Lab2

This is the function which will be called in the main function which will accumulate the decimal number from the char array and place it in the integer variable decimal.

C source code appended to file lab.c

```
void accumulateDecimalNumber()
{
    decimal = 0;
    i = 0;
    adnwhileLoop:
        getdecimalDigit();
        decimal *= 10;
        decimal += decDigit;
        if(i >= numberOfDecDigits−1) //start
        {
            goto adnendwhileLoop;
        }
        i++;
        goto adnwhileLoop;
    adnendwhileLoop:
```

- All comments go here

ASM source code appended to file lab.s

```
accumulateDecimalNumber: #procedure
    mov $0 , decimal
    mov $0, i
    adnwhileLoop:
    call getdecimalDigit
    mov decimal, %eax #stores decima
    imul $10 #multiply by 10
    addl decDigit, %eax #adds decDig
    mov numberOfDecDigits, %ebx
    dec %ebx
    cmp i,%ebx
    jle adnendwhileLoop
    inc i
    jmp adnwhileLoop
    adnendwhileLoop:
    ret
```

CS118 Lab2

This is the function which will calculate the number of hexadecimal digits required for this conversion.

C source code appended to file lab.c

```
void numberOfHexDigits()
    temp = decimal;
    numHexDigits = 0;
    nhdwhileLoop:
    numHexDigits++;
    if(temp == 0)
    {
        goto endwhileLoop;

    }
    temp = temp/16;
    goto nhdwhileLoop;
    endwhileLoop:
    return;
}
}
```

- All comments go here

ASM source code appended to file lab.s

```
numberOfHexDigits: #procedure 5
    mov decimal, %eax
    mov $0, numHexDigits
    nhdwhileLoop:
    inc numHexDigits
    cmp $0, %eax #eax is temp
    je endwhileLoop
    mov $0, %edx
    idiv $16
    jmp nhdwhileLoop
    endwhileLoop:
    ret
```

CS118 Lab2

This is the function which will calculate the number of hexadecimal digits required for this conversion.

C source code appended to file lab.c

```
void convertDectoHex()
{
    numHexDigits = 0;
    numberOfHexDigits();
    i = numHexDigits−1;
    d2hwhileLoop:
    gethexDigit();
    hexDigits[i] = hexarray[hexDigit];
    if(i <= 0)
    {
        goto d2hendwhileloop;
    }
    i−−;
    goto d2hwhileLoop;
    d2hendwhileloop:
    return;
```

- All comments go here

ASM source code appended to file lab.s

```
convertDectoHex: #procedure 6
    mov $0, numHexDigits
    call numberOfHexDigits
    mov numHexDigits, %eax
    sub $1, %eax
    d2hwhileLoop:
    call gethexDigit
    mov hexDigit, %ebx #line 69
    mov $hexarray, %esi #stores strin
    mov i, %ecx
    mov $hexDigits, %edi #line 69 (d
    cmp $0, %eax
    jle d2hendwhileloop
    dec i
    jmp d2hwhileLoop
    d2hendwhileloop:
```

Build script

Text written to file build.sh

```
doctex lab
pptexenv latex lab
dvipdf lab
gcc lab.c -o labc -lm
#gdb -quiet ./labc
```

Bourne Shell

```
chmod +x build.sh
```