CS 3844 Computer Organization
Term Project Part 2 – Spring 2021

Member Name/abc123: Thomas White/yhy312

1. Using Visual Studio on the UTSA VDI system, create a project using the code given below.

```
unsigned char     gArray[]    = { 0x09, 0xFA, 0x5A, 0x18, 0x48, 0xAC, 0xD4, 0x71 };
short int         gArraySI[]  = { 0x09, 0xFA, 0x5A, 0x18, 0x48, 0xAC, 0xD4, 0x71 };
int               gArrayI[]   = { 0x09, 0xFA, 0x5A, 0x18, 0x48, 0xAC, 0xD4, 0x71 };

int test()
{
      __asm {
            mov al,gArray                 al = \t
            lea esi,gArray                esi= 11100160
            mov dl, byte ptr [esi]        dl = \t
            mov edx, dword ptr [esi+2]    edx= 2890405978
            mov al,gArray[5]              al = ¬
            nop
            mov ax,gArraySI               ax = 9
            lea esi,gArraySI              esi= 11100168
            mov dl, byte ptr [esi]        dl = \t
            mov edx, dword ptr [esi+2]    edx= 5898490
            mov ax,gArraySI[5]            ax = 6144
            nop
            mov eax,gArrayI               eax= 9
            lea esi,gArrayI               esi= 11100184
            mov dl, byte ptr [esi]        dl = \t
            mov edx, dword ptr [esi+2]    edx= 16384000
            mov eax,gArrayI[5]            eax= 1509949440
            nop
      }
      return 0;
}

int main( int argc, char *argv[] )
{
      test();
}
```

2. Compile and single step through the program, writing down the value of the destination for each instruction.
(50 pt)

3. Using the hen machines, use gdb to trace the assembly code in the executable file *Part2Bonus*. The following is the part of the assembly for the function *func1*. Fill in the instructions for each line in the left column of blank lines and enter the value in the register/address indicated on the blank lines of the right column. If a line was not executed, enter the instruction in the left column and write "did not execute" on the corresponding line in the right column.
(50 pt)

Note: the hex addresses will probably be different when you run gdb, but the addresses of the form <func1+#> should be the same.

func1:

```
0x08048394 <func1+0>:    push   %ebp
0x08048395 <func1+1>:    mov    %esp,%ebp
0x08048397 <func1+3>:    sub    $0x10,%esp
0x0804839a <func1+6>:    mov    %ebx,(%esp)
0x0804839d <func1+9>:    mov    %esi,0x4(%esp)
```

0x080483a1 <func1+13>:	mov 0x8(%ebp), %edx	%edx: 3
0x080483a4 <func1+16>:	mov 0xc(%ebp), %ecx	%ecx: 0xbffff6a0
0x080483a7 <func1+19>:	mov (%ecx,%edx,4),%eax	%eax: 4
0x080483aa <func1+22>:	mov 0x10(%ebp),%ebx	%ebx: 0xbffff68c
0x080483ad <func1+25>:	mov (%ebpx,%edx,4),%esi	%esi: 16
0x080483b0 <func1+28>:	cmp %esi,%eax	%eax: 4
0x080483b2 <func1+30>:	jle 0x80483b9	
0x080483b4 <func1+32>:	mov %eax,(%ebx,%edx,4)	(%ebx,%edx,4): 0xbffff698
0x080483b7 <func1+35>:	jmp 0x80483be	
0x080483b9 <func1+37>:	mov %esi,(%ecx,%edx,4)	(%ecx,%edx,4): 0xbffff6ac
0x080483bc <func1+40>:	mov %esi,%eax	%eax: 16

```
0x080483be <func1+42>:   pop    %ebx
0x080483bf <func1+43>:   pop    %esi
0x080483c0 <func1+44>:   add    $0x8,%esp
0x080483c3 <func1+47>:   leave
0x080483c4 <func1+48>:   ret
```

4. Given the following main function, fill in the arguments (there is more than one) to the call to *func1* that would produce the assembly code above. (Bonus 5 pt)

```
int main(){
   int array1[] = {10, 12, 3, 4, 25};
   int array2[] = {9, 28, 7, 16, 5};

      func1(_____);
}
```

5. Write a function in c, named *func1*, that duplicates the functionality of the above assembly code: (**Bonus 5 pt)**

6. **Each team member needs to submit all files in ProjectP2.zip with names and abc123 of all team members to BB,**