INSTRUCTIONS: Show your work (i.e., how you derived your answer or the reason behind your thinking) in addition to your answer. **Budget your time wisely** (e.g., do not spend too much time on a single question). You have 110 minutes to work on this exam. The total number of points is 100.

1. (5 points) An Apple II computer uses 16-bit addresses to access the memory. Each address is transmitted twice on the 8-bit address bus.
   (a) What is the maximum size of memory (in bytes) it can support?
   (b) If each address were transmitted only once on the address bus (to form a 8-bit memory address), then what would be the maximum supported memory size?

2. (5 points) Which two of the following produce the same 4 bytes in MASM?
   
   A1 BYTE “1234”
   A2 BYTE 1,2,3,4
   A3 BYTE 4,3,2,1
   A4 WORD 1,2,3,4
   A5 DWORD 01020304H
   A6 DWORD 01020304
   A7 DWORD 04030201

3. (5 points) Write an instruction that moves the first two bytes in the following array (A1) to the AX register. The resulting value will be 2010h.
   
   A1 BYTE 10h, 20h, 30h, 40h

4. (7 points) Suppose there were no `PUSH` instruction. Write a sequence of two other instructions that would accomplish the same as `PUSH EAX`.

5. (8 points) In the following instruction sequence, show the changed value of AL where indicated, in hexadecimal:
   
   MOV AL, 01101111b  
   AND AL, 00101101b  ; (a)  
   MOV AL, 6Dh  
   AND AL, 4Ah  ; (b)  
   MOV AL, 00001111b  
   OR AL, 61h  ; (c)  
   MOV AL, 94h  
   XOR AL, 37h  ; (d)  

6. (5 points) A limitation of the `LOOP` instruction is that it must jump to a label that is −128 to +127 bytes from the current location. That is why MASM shows the “jump destination too far” error if the loop body is too long. However, the `Jcondition` (Conditional Jump, e.g., `JE`, `JG`, `JL`...etc.) instruction does not have such a limitation. Show how to replace the `LOOP` instruction in a loop that contains a large loop body.
7. (5 points) Identify the problems in the following code and propose a solution to fix it.

```assembly
main PROC
    call Example1
    exit
main ENDP
Example1 PROC
    push 5
    push 6
    call AddTwo
    ret
Example1 ENDP
Stub PROC
    ; do nothing
    ret
Stub ENDP
```

8. (3 points) How many times will the following loop execute?

```assembly
X2: mov ecx, 0
    inc ax
    loop X2
```

9. (7 points) Create a macro named `mAdd16` that adds any two signed 16-bit memory operands and produces a 16-bit sum. Remember to save the content of the register before you use it for addition, and restore it when you are done. Syntax: `mAdd16 sum, op1, op2`

10. (5 points) Write instructions that jump to label L4 if bits 1, 2, and 3 are all set in the DL register. (Note: the right-most bit is bit 0.)

11. (10 points) Use the following data definitions for this question.

```assembly
dArray DWORD 10 DUP(?)
dSize = ($ - dArray)
byte1 BYTE 0FFh, 1, 2
word3 SWORD 7FFFh, 8000h
```

Where marked by a letter (a, b, c, d, e, f, g, h) in the following code segment, give your answer and explain your reasons. Suppose the code segment is executed sequentially from top to bottom. Note that some instructions may be illegal.

```assembly
mov ax, dSize  a. ax = ?
mov ax, [word3+2]  b. ax = ?
mov eax, [word3+4]  c. eax = ?
mov OFFSET byte1, 10h  d. byte1 = ?
mov ebx, OFFSET byte1
mov al, [ebx+3]  e. al = ?
movsx eax, byte1  f. eax = ?
mov al, 80h  g. ZF, CF, SF, OF = ?
add al, 80h  g. ZF, CF, SF = ?
mov al, 00110011b  h. ZF, CF, SF = ?
test al, 2
```
12. (10 points) Use jl and jg to implement the following pseudo code.

   while( int2 >= int1 ){
      add ebx,2
      if( ebx > int2)  
         mov ebx,0
      else
         mov ebx,int1
   }

13. (15 points) Implement the following pseudo code and return the sum in EAX. Use explicit stack parameters (such as [ebp+n]) and follow common procedure conventions (e.g., stack frame). Draw a figure to show the contents of the stack right before return, including the ebp and esp.

   int ProcTwo(int x, y)
   { int i = 5;
     Return x + y + i;
   }

14. (10 points) To evaluate a compound Boolean expression, we typically use Short-Circuit Evaluation. For examples, the second part of the AND condition in the following code is not evaluated and b is still 1. However, some languages (e.g., BASIC) use Non-Short-Circuit Evaluation, which will cause b to become 0. Implement the following code in assembly using the Non-Short-Circuit Evaluation.

   a = 10;
   b = 1;
   if((a > 0) && (b-- > 0)) {
      c = a;
   }
   // Is b 1 or 0 now?