Register Indirect addressing is defined as follows:

- Accessing register contents as a value.
- Accessing memory through an address stored in a register.
- Accessing a memory area specified and maintained by a pointer in the ESP register.
- None of these.

The RET instruction (without operands) will pop how many bytes off the stack?

- 8
- 16
- 2
- 4

When passing procedure parameters on the stack, why are the following lines of code often necessary in a procedure?

```
push  ebp
mov   ebp,esp
```

- Because the procedure might change the EBP register value.
- To preserve the original EBP register value for register indirect addressing.
- They are never necessary.
- To keep additional usage of the stack within the procedure from invalidating the stack offsets.

The following two instructions are equivalent.

```
ret
ret 4
```

- True
- False

Given the following register states, and using Base Indexed Addressing, which of the following lines of code will move the 11th element of the *list* array (of DWORDs) to the EAX register?

EDX register contans the address of the first element of *list*.
ESI register contains the address of the eleventh element of *list*.
EBX register contains the value 40,

- mov eax, list[esi]

- [x] mov eax, [edx + ebx]
- [ ] mov eax, list[ebx]
- [ ] mov eax, [esi]

Given the following register states, and using Register Indirect Addressing, which of the following lines of code will move the 11th element of the *list* array (of DWORDs) to the EAX register?

EDX register contans the address of the first element of *list*.
ESI register contains the address of the eleventh element of *list*.
EBX register contains the value 40,

- [ ] mov eax, [edx + ebx]
- [ ] mov eax, list[ebx]
- [x] mov eax, [esi]
- [ ] mov eax, list[esi]

Given the following register states, and using Indexed Addressing, which of the following lines of code will move the 11th element of the *list* array (of DWORDs) to the EAX register?

EDX register contans the address of the first element of *list*.
ESI register contains the address of the eleventh element of *list*.
EBX register contains the value 40,

- [x] mov eax, list[ebx]
- [ ] mov eax, [esi]
- [ ] mov eax, [edx + ebx]
- [ ] mov eax, list[esi]

If you reference a point beyond the end of an array in MASM (for example, the address of the what would be the 105th element of a 100-element array), what happens?

- [ ] The disassembler prevents your program from compiling.
- [ ] Run-time error
- [ ] Compile-time error
- [x] You attempt to access whatever data bytes are stored there.

Suppose that you are given the following program (with memory addresses shown on the left).
Inside *someProcedure*, what numerical operand should be used with the *RET* instruction?

```
.data
```

```
x   DWORD  153461
y   BYTE   37
z   BYTE   90


.code
main PROC
push  x
push  y
push  z
call  someProcedure
inc   EAX
mov   EBX, z
xor   EAX, EBX
exit
main ENDP
END MAIN
```

6

Suppose that you are given the following program (with memory addresses shown on the left). Inside *someProcedure*, what numerical operand should be used with the *RET* instruction?

```
.data
x   DWORD  153461
y   BYTE   37
z   BYTE   90


.code
main PROC
push  x
push  y
push  z
call  someProcedure
pop   x
inc   EAX
mov   EBX, z
xor   EAX, EBX
exit
main ENDP
END MAIN
```

2

Suppose that you are given the following program (with memory addresses shown on the left). What hexadecimal value does EIP hold immediately after "inc EAX" has executed?

```
.data
0x100   x   DWORD  153461
0x104   y   BYTE   37
0x105   z   BYTE   90


.code
main PROC
0x12   push  x
0x17   mov   AH, y
0x1C   mov   AL, z
0x21   call  someProcedure
0x26   inc   EAX
0x2B   mov   EBX, z
0x30   xor   EAX, EBX
0x35   exit
main ENDP
END MAIN
```

2B

For this problem, suppose that you are working with the partial data segment given below. Assume that the memory address of **balance** is 0x44. What hexadecimal address belongs to the **first** item in **history**?

```
HISTLIMIT = 100

.data
balance   DWORD   0
account   WORD    ?
history   WORD    HISTLIMIT DUP(?)
isValid   BYTE    0
```

4A

Given the following partial data segment, what value would I put in the brackets in list$[n]$4 to access the 15th element of *list*? (Ignore the .0000 that Canvas may append to your answer).

```
.MAX = 50
.data
list  DWORD    MAX   DUP(0)
a     DWORD    25
b     DWORD    15
```

56

The RET instruction pops the top of the stack into what register?

- ESP
- EBP
- It does not pop the top of the stack into a register.
- EIP

Arrays are stored in _____ memory.

- Random
- Contiguous
- Disjoint

Given *list*, an array of WORDs, what element is addressed by *list[8]*?

Hint: It's Love.

- 8th Element
- 4th Element
- 5th Element
- 9th Element

The following instruction will increment the stack pointer (ESP) by how many bytes?

```
ret 8
```

12

Suppose that you are given the following program (with memory addresses shown on the left).
After the instruction "mov ebp, esp", which of the following is referenced by each of the following?

```
.data
x   DWORD  153461
y   WORD   37
z   WORD   90


.code
main PROC
   push  x
   push  y
   push  z
   call  someProcedure
   ...
   exit
main ENDP

someProcedure PROC
   push ebp
   mov ebp, esp
   ...
   pop ebp
   ret 8
someProcedure ENDP
END MAIN
```

[ebp + 4] The return address from someProcedure.

[ebp + 8] The decimal value 90.

[ebp + 10] The decimal value 37.

[ebp + 12] The decimal value 153461.

[ebp] The previous value of EBP.

[ebp + 6] None of these.

Given the following partial data segment, what value would I put in the brackets in list[n]6 to access the 8th element of *list*? (Ignore the .0000 that Canvas may append to your answer).

```
.MAX = 50
.data
```

```
list  DWORD    MAX   DUP(0)
a     DWORD    25
b     DWORD    15
```

28

For this problem, suppose that you are working with the partial data segment given below. Assume that the memory address of **balance** is 0x44. What hexadecimal address belongs to the**last** item in **history**?

```
HISTLIMIT = 100

.data
balance   DWORD   0
account   WORD    ?
history   WORD    HISTLIMIT DUP(?)
isValid   BYTE    0
```

110h