Rico Adrian

CS 311

Lab 03

Computation

1. a. memory displacement = 00000000. by default, first line always starts from zero. Then, the memory displacement in line b will be based on the input of column a, c will be based on the input of column b, etc.

b. memory displacement = 00000001. Increase by 1 (in base 10) because the data type is db and there is only 1 value stored in the field. (1\*1 byte).

c. memory displacement = 000000003. Increase by 2 (in base 10) because the data type is dw and there is only 1 value stored in the field. (1\*2 byte).

d. memory displacement = 00000007. Increase by 4 (in base 10) because the data type is dd and there is only 1 value stored in the field. (1\*4 byte).

e. memory displacement = 0000000F. Increase by 8 (in base 10) because the data type is dq and there is only 1 value stored in the field. (1\*8 byte). Calculation: 7+8 = 15 is equal to F in base 16.

f. memory displacement = 00000025. Increase by 22 (in base 10) because the data type is dw and there are 2 values stored in this field. The first one is string with 20 characters and the second one is the value 0. The calculation of memory on this one would be (20\*1 byte + 1\*2 byte = 22 byte). In the case of String, the data type (db, dw, dd, dq) does not matter on the memory displacement. Therefore, even though the data type is dw, it’s memory is 20\*1 byte instead of 20\*2 byte. Meanwhile, for the second value, 0, the memory is 1\*2 byte, just like usual. Therefore, the final result is 15+22 =37(in base 10). After the conversion, it would be 00000025 in base 16.

g. memory displacement = 00000029. Increase by 4 (in base 10) because the data type is dd and there is 1 only value stored in this field. (1\*4 byte). Calculation: 37+4=41(in base 10) = 00000029(in base 16).

h. memory displacement = 00000065. Increase by 60 (in base 10) because the data type is dw and there are 30 values stored in this field (an array of 30 dw). (30\*2 byte). Calculation: 41+60=101(in base 10) = 00000065(in base 16). (6\*16^1 + 5\*16^0 = 101). Note: the memory displacement for the last column (dd 0xa46b0) does not matter because we always discard the last one since we started on 0, not column a.

1. i. memory displacement = 00000000. by default, first line always starts from zero. Because it’s a new segment, it starts on 0 again instead of continuing the column h. Then, the memory displacement in line j will be based on i, and line k will be based on j.

j. memory displacement = 000000064. Increase by 100 (in base 10) because the data type is resd and there are 25 values stored in the field (an array of 25 dd). (25\*4 byte). Calculation: 0 + 100 = 100(in base 10) = 00000064(in base 16). (6\*16^1 + 4\*16^0=100).

k. memory displacement = 0000000C8. Increase by 100 (in base 10) because the data type is db and there are 100 values stored in the field (an array of 100 db). (100\*1 byte). Calculation: 100 + 100 = 200(in base 10) = 000000C8(in base 16). (C=12).

(12\*16^1 + 8\*16^0=200). Note: Again, the memory displacement for the last column (resw 20) does not matter because we always discard the last one since we started on 0, not column i.