HW8

8.1, 8.12 AND 8.16

8.1 Suppose we are compiling for a machine with 1-byte characters, 2-byte shorts, 4-byte integers, and 8-byte reals, and with alignment rules that require the address of every primitive data element to be an even multiple of the element's size. Suppose further that the compiler is not permitted to reorder fields. How much space will be consumed by the following array? Explain.

```
A : array [0..9] of record
        s : short
        c : char
        t : short
        d : char
        r : real
        i : integer
```

s : + 2
c : + 1
+ 1 (padding for short)
t : + 2
d : + 1
+ 2 (padding for real)
r : + 8
i : + 4
+ 3 (end padding
= 24
24 * 10 = 240

240 bytes

8.12 Consider the following C declaration, compiled on a 64-bit x86 machine:

```
struct {
    int n;
    char c;
} A[10][10];
```

If the address of `A[0][0]` is 1000 (decimal), what is the address of `A[3][7]`?

+ 4 int
+ 1 char
+ 3 padding
= 8

A[10][10] is a 2D array with each element being one struct.

The address for any element A[i][j] can be calculated using the formula: base address + (i * size of one array of structs + j * size of one struct). Since the base address of A[0][0] is 1000 and we are looking for the address of A[3][7], we calculate the offset from the base address as follows:

offset = 1000 + (3 * (8 * 10) + 7 * 8)
= 1296

8.16 Explain the meaning of the following C declarations:

```
double *a[n];
double (*b)[n];
double (*c[n])();
double (*d())[n];
```

double *a[n];

This declares a as an array of n pointers to double. Each element in a is a pointer that can point to a double value.

double (*b)[n];

This declares b as a pointer to an array of n double values. Here, b is a pointer that points to a whole array of n elements of type double.

double (*c[n])();

This declares c as an array of n pointers to functions that take unspecified arguments and return a double. Each element of c can point to a function with the mentioned signature.

double (*d())[n];

This declares d as a function that takes no

arguments and returns a pointer to an array of n double values. The function d when called will return such a pointer.