Akhil Cherukuri
CMPE 244
LAB 1: Struct Address

Q) http://books.socialledge.com/books/useful-knowledge/page/struct-address

A)

To Do: add int main(void) {my_s s; ….. }

With-out Packed Attribute (Padded Struct):

```
entry_point(): Entering main()
Size : 16 bytes
floats 0x0x1000ffd0 0x0x1000ffd8
chars  0x0x1000ffd4 0x0x1000ffdc
Starting RTOS
```

With Packed Attribute (Packed Struct):

```
entry_point(): Entering main()
Size : 10 bytes
floats 0x0x1000ffd4 0x0x1000ffd9
chars  0x0x1000ffd8 0x0x1000ffdd
Starting RTOS
```

Observation:

1) Size of Packed Struct is 6 bytes less than Padded Struct for a function of 2 floats (2*4 bytes=8) and 2 chars (2*1 byte = 2), mathematically added up should give a size of 10 bytes which is in the case of Packed Struct.

   Packed Struct = [Float(4bytes) + Char(1byte) +Float(4bytes) + Char(1byte) = 10 bytes]
   Padded Struct = [Float(4bytes) + Char(1byte + 3bytes padding) + Float(4bytes) + Char(1byte + 3bytes padding) = 16 bytes]

2) We can observe that in Padded Struct, compiler will introduce alignment requirement to every structure. It will be as that of the largest member of the structure. This is done to prevent confliction between alignment requirements with various data type members by aligning structure members to natural address boundaries.

3) Packing will make memory access slower than Padding.