Recitation#5: Reinforcements on code execution and memory layout

CS232 Spring 2021

When: February 26 at 2:00 pm

Puzzlers for Pointers, Addresses, and Values
A memory has the following contents (in little-endian format) MOST SIGNIFICANT is at Highest least significant at lowest

Variable	Address	Bytes	Final Value of Byte
A	0x08000000	**00 00 00 08**	**0c 00 00 08**
B	0x08000004	**04 00 00 08**	
C	0x08000008	**fe ff ff ff**	**00 00 00 00**
D	0x0800000C	**ff ff ff ff**	**02 00 00 00**
E	0x08000010	**00 00 00 00**	**05 00 00 00**
F	0x08000014	**01 00 00 00**	**18 00 00 08**
G	0x08000018	**02 03 04 05**	
H	0x0800001C	**33 35 31 00**	**04 00 00 08**

Given the following declarations (assuming a 32-bit architecture):

```
int *A, *B; float C; int D; float E; int F; float G;

typedef struct xform {
    int i[2][2];
    float * factor;
    int color;
} xform;

xform *H;
```

Fill in columns for the address (in hex) that is changed in each statement and the value (in hex) to which it is changed. **NOTE: The statements are executed in sequence and changes made to memory apply in the following lines**.

C statements	Address(hex)	Value(hex)
`A = B + 2;`	0x08000000	**0x0800000c**
`C = (float) (*A + F);`	0x08000008	**0x00000000**
`H = (xform *) &B;`	0x0800001c	**0x08000004**
`H->factor = &E + 2;`	0x08000014	**0x08000018**
`D = (int) *((char *)(H->factor));`	0x0800000c	**0x00000002**
`H->i[(D >> 1)][1] = D + 3;`	0x08000010	**0x00000005**