Willis T. Allstead
Professor Egbert
CPE 301
28 September 2016

HW 3

1)

a) volatile int *y_addr;
b) volatile unsigned char *ch_addr;
c) volatile int *z;
d) volatile int *date_pt;
e) volatile unsigned char *pt_chr;

2)

a) valid
b) invalid
c) invalid
d) invalid
e) invalid
f) invalid
g) invalid
h) valid
i) invalid
j) invalid
k) invalid
l) invalid
m) invalid
n) valid
o) invalid
p) invalid
q) invalid
r) invalid
s) valid
t) invalid
u) invalid

3)

An address must be stored.

4)

&var2 means the address of var2.

5)

On my macOS system:

Size of address of an int is 8
Size of address of a char is 8
Size of address of a double is 8
Size of an int is 4
Size of a char is 1
Size of a double is 8

I would have expected the size of each address to be the same since it is only used to access the data at that address, more bits are needed for different types.

6)

a) 10000000
b) 11101111
c) 01101111

7)

a) $(10000000)_2$ -> $(80)_{16}$
b) $(11101111)_2$ -> $(EF)_{16}$
c) $(01101111)_2$ -> $(6F)_{16}$

8)

a) 0x0157 = 0b0000,0001,0101,0111
shifted left 1: 0b0000,0010,1010,1110
0b0000,0010,1010,1110 = 0x02AE

b) 0x0701 = 0b0000,0111,0000,0001
shifted left 2: 0b0001,1100,0000,0100
0b0001,1100,0000,0100 = 0x1C04

c) 0x0673 = 0b0000,0110,0111,0011
shifted right 2: 0b0000,0001,1001,1100
0b0000,0001,1001,1100 = 0x019C

d) 0x0057 = 0b0000,0000,0101,0111
shifted right 3: 0b0000,0000,0000,1010
0b0000,0000,0000,1010 = 0x000A

9)

a) supplied int: xxxxxxxx
mask: 00001100
0b0000,1100 = 0x0C

b) supplied int: xxxxxxxx
mask: 10101111
0b1010,1111 = 0xAF

```
     c) supplied int: xxxxxxxx
                mask: 00000101 // This will be an XOR operation
        0b0000,0101 = 0x05
10)

unsigned char *p = (unsigned char *)0x417;
(*p) |= 0x40; // 0b0100,0000 in hex form


or if bits are ordered least->most significant:

(*p) |= 0x02; // 0b0000,0010 in hex form
```