CS 354 Midterm Review

C Programming

1. Basics - functions
2. Data types
3. Type casting
4. Arrays
5. Pointers
6. Pointer to a pointer
7. Pointers and Arrays
8. Type casting pointers
9. Structures
10. Linked List

Data Representation

1. Bitwise Operators in C - &, |, ~, ^
2. Logical Operators in C - &&, ||, !
3. Little Endian vs Big Endian
4. 1's complement, **2's complement**, and signed magnitude
5. Integer Arithmetic
    a. Addition
    b. Subtraction
    c. Multiplication (using left shifting)
    d. Division (using right shifting)

Assembly

1. Registers
2. MOV instruction - - movsbw, movzbl, etc.
3. Data Formats - b, w, l
4. Operand Forms
    a. Immediate
    b. Register
    c. Memory - e.g. 0x480032
    d. Memory - Scaled Indexed

Review

```
int main()
{
    ____
    ____
    ____
    return 0;
}
```

```
int func(args)
{
    ____
    ____
    ____
}
```

args - passed by value!

```
func(int n)
{
    ____
    ____
}
```

n is local to func.

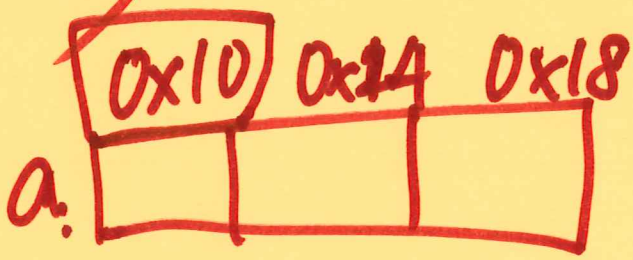
```
int l=10;
func(l);
```

```
func(int a[])
```

```
func(a);
```

a-arra

func (&a[0])

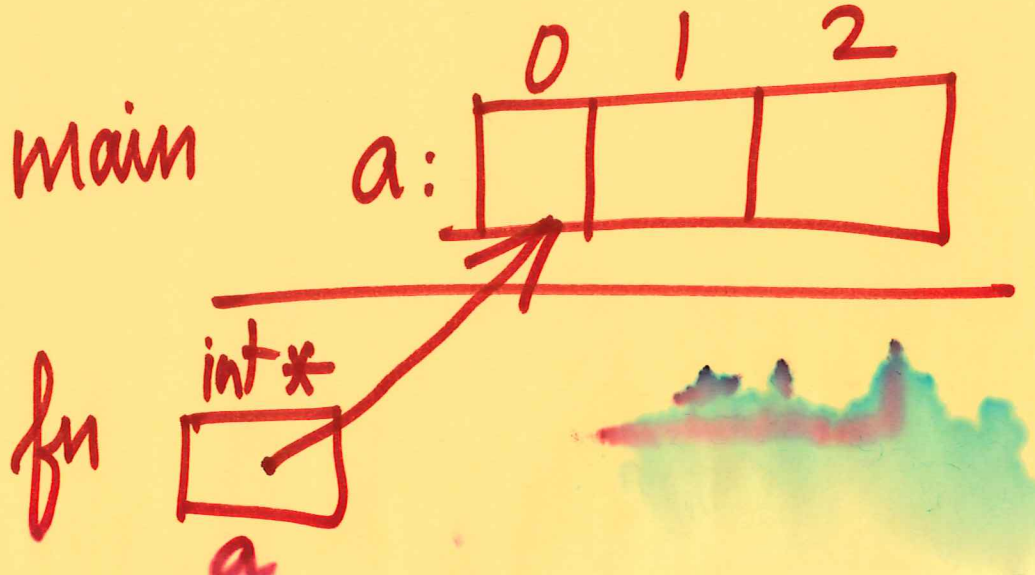


func (int *a)

(or)

int a[]

0x10 - &a[0]



~~int~~
~~short~~
~~char~~

char - 1 byte
short int - 2 bytes
int - 4 bytes.
long int - 4 or 8 bytes.

~~float~~
~~double~~ X

int X = 0x ABCD12

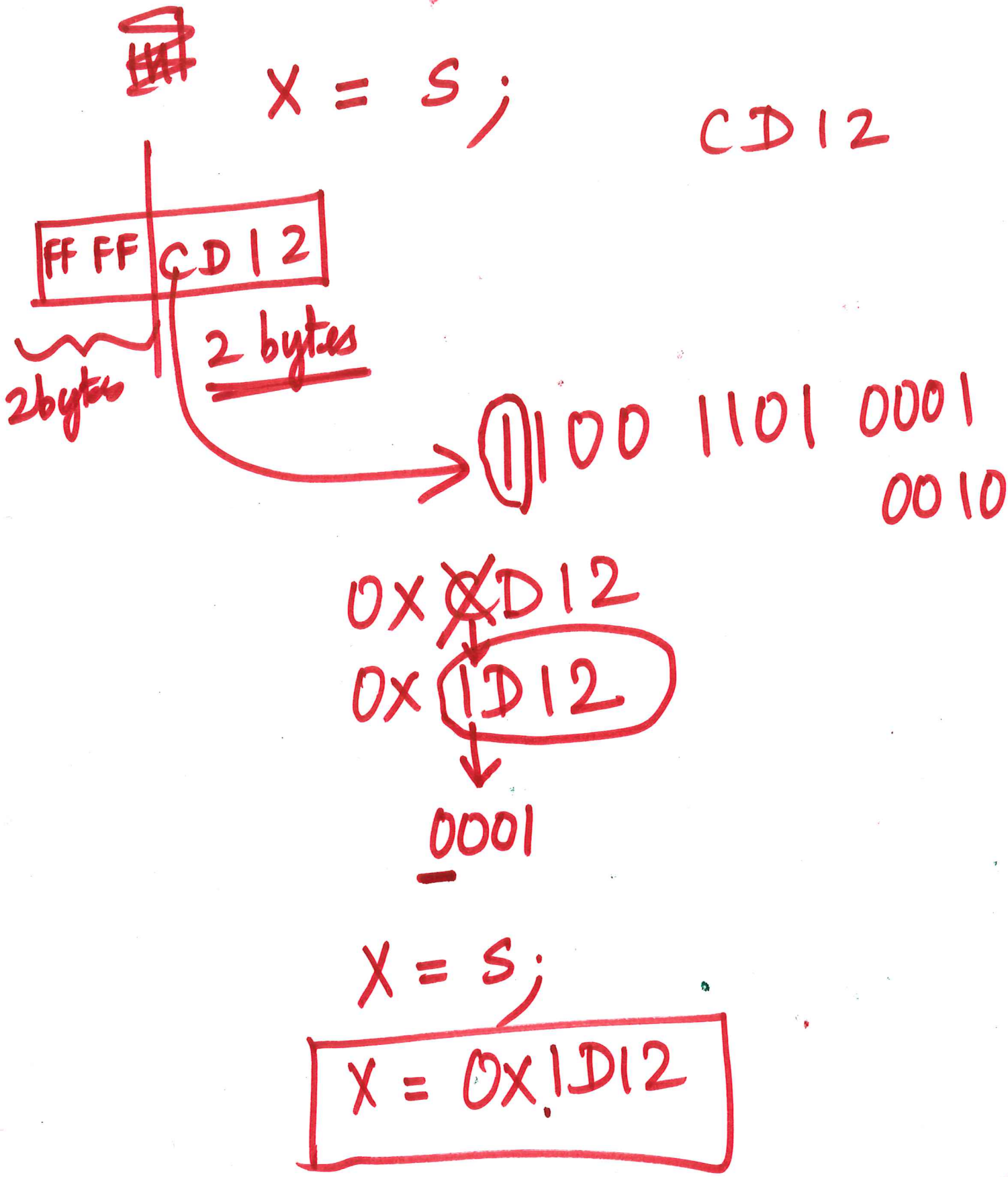
short S = X;

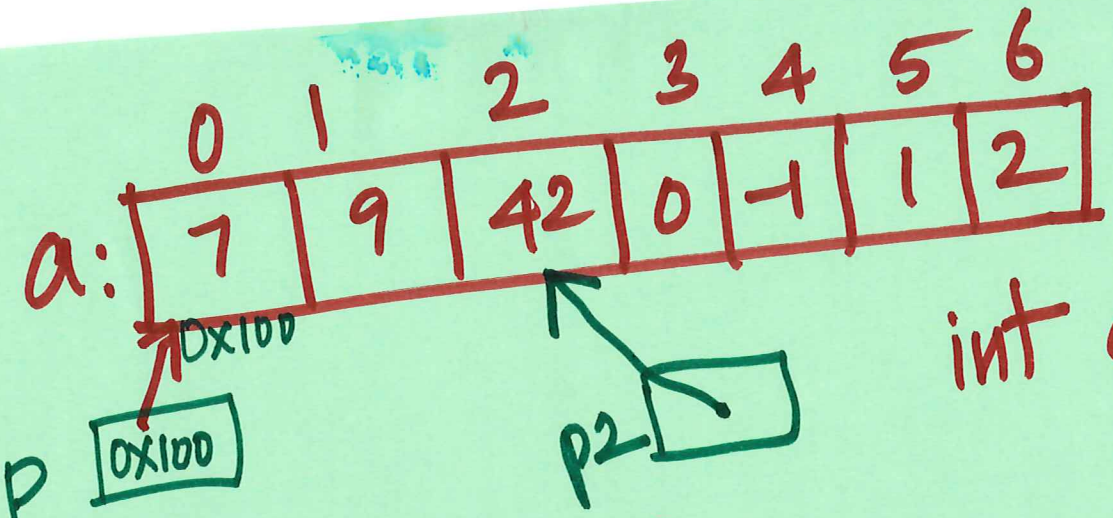
1 bytes
3 bytes

0x ~~00~~ ABCD12

short S = (short) X;

S
0x CD12


X = S;
CD12
FF FF CD12
2 bytes
2bytes
1100 1101 0001
0010
0X CD12
0X 1D12
0001
X = S;
X = 0X1D12

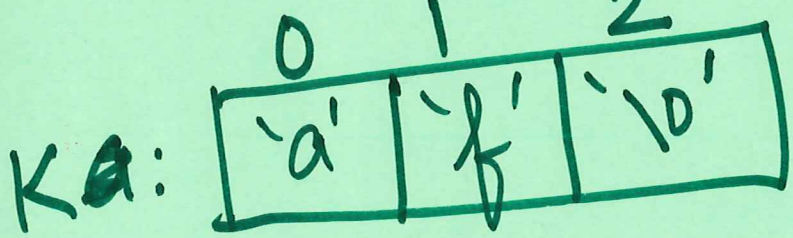


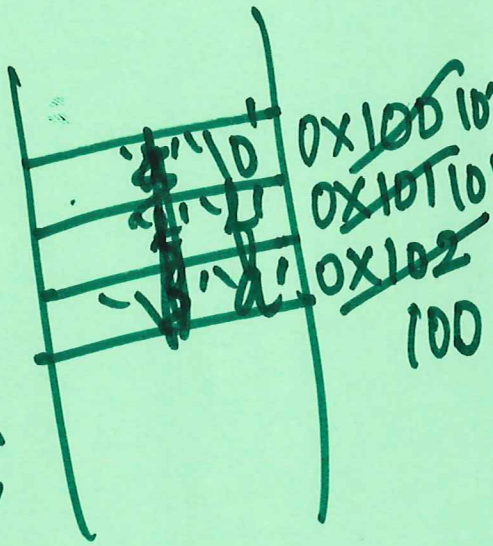
int a[7];

int *p = a;

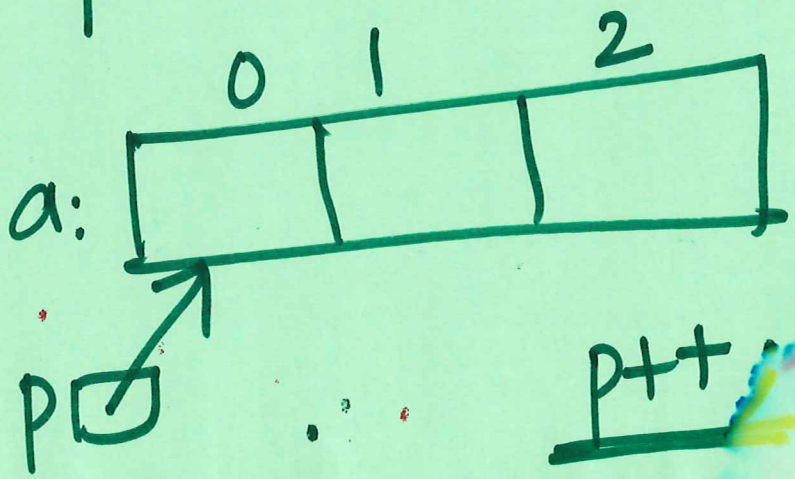
= &a[0];

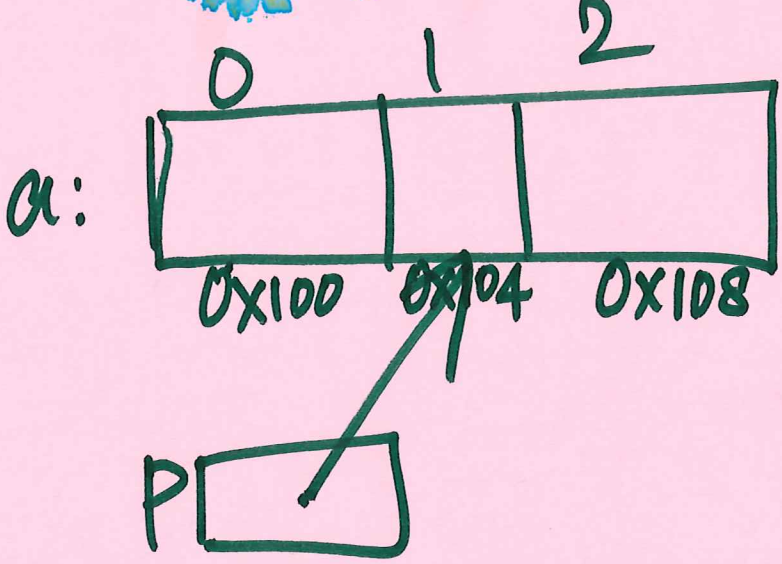
char K[3];





int *p2 = &a[2];

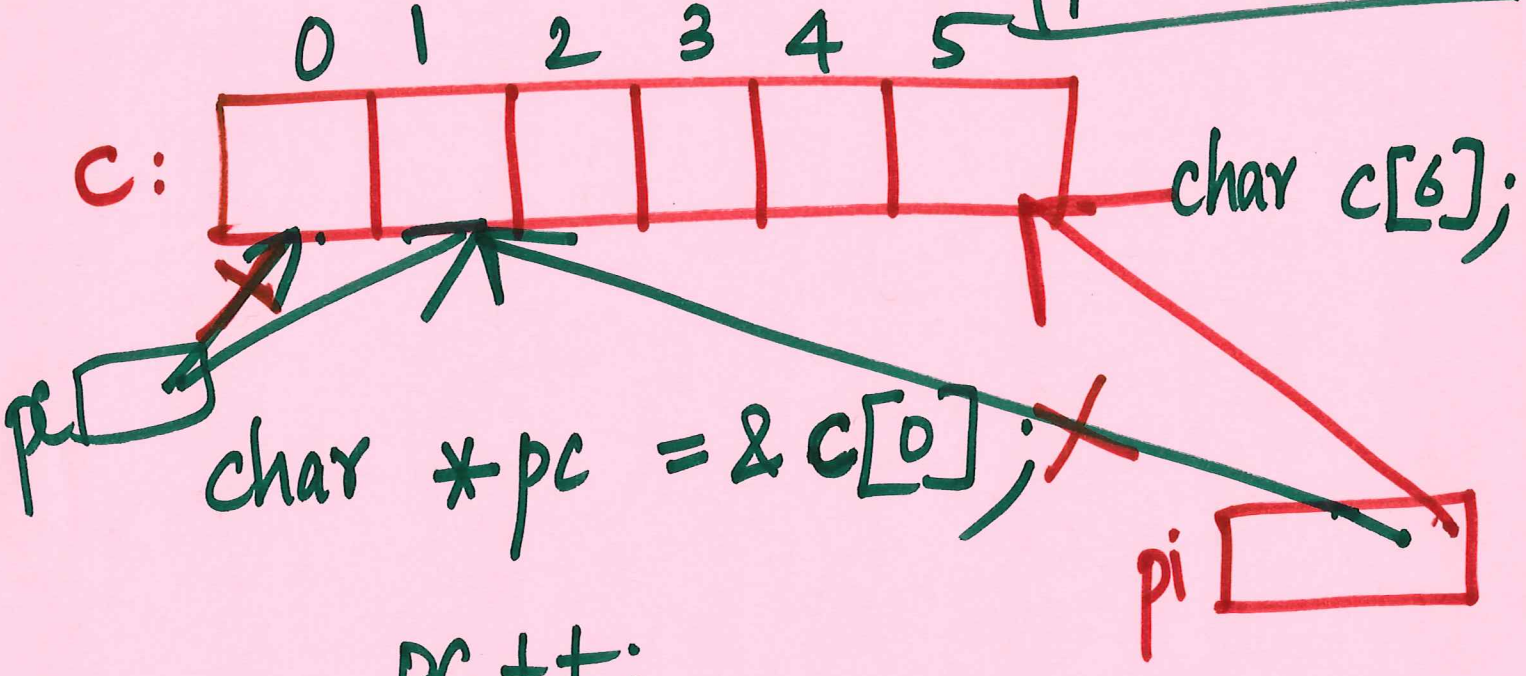




P = P + 1;

= 0x100 + 4

P = 0x104



pc++;

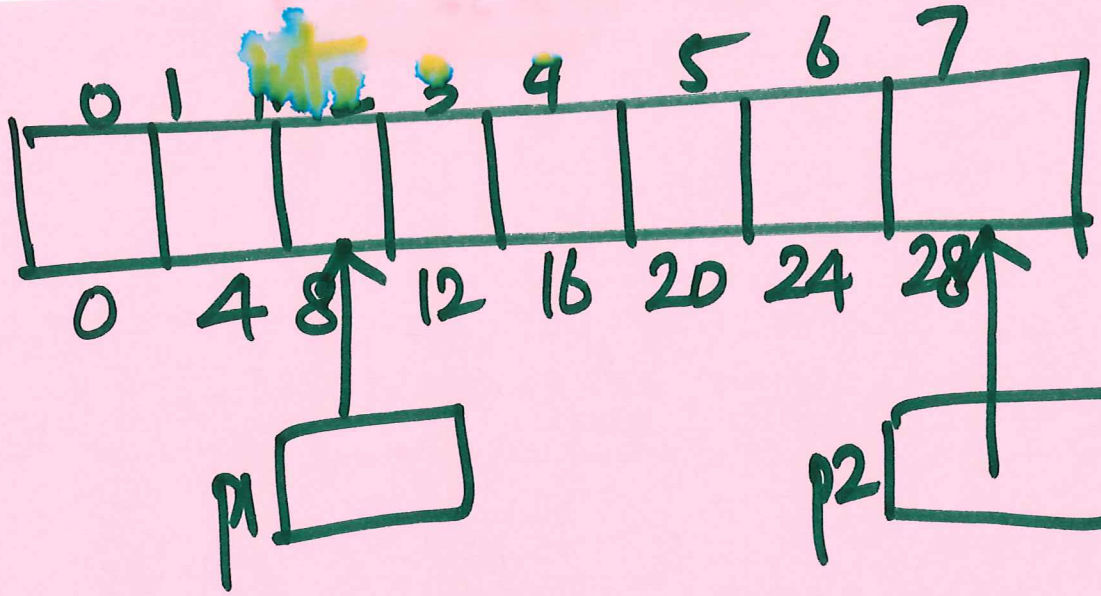
int *pi = (int*) pc;

pi++;

pi++;

p = &c[5]

a:



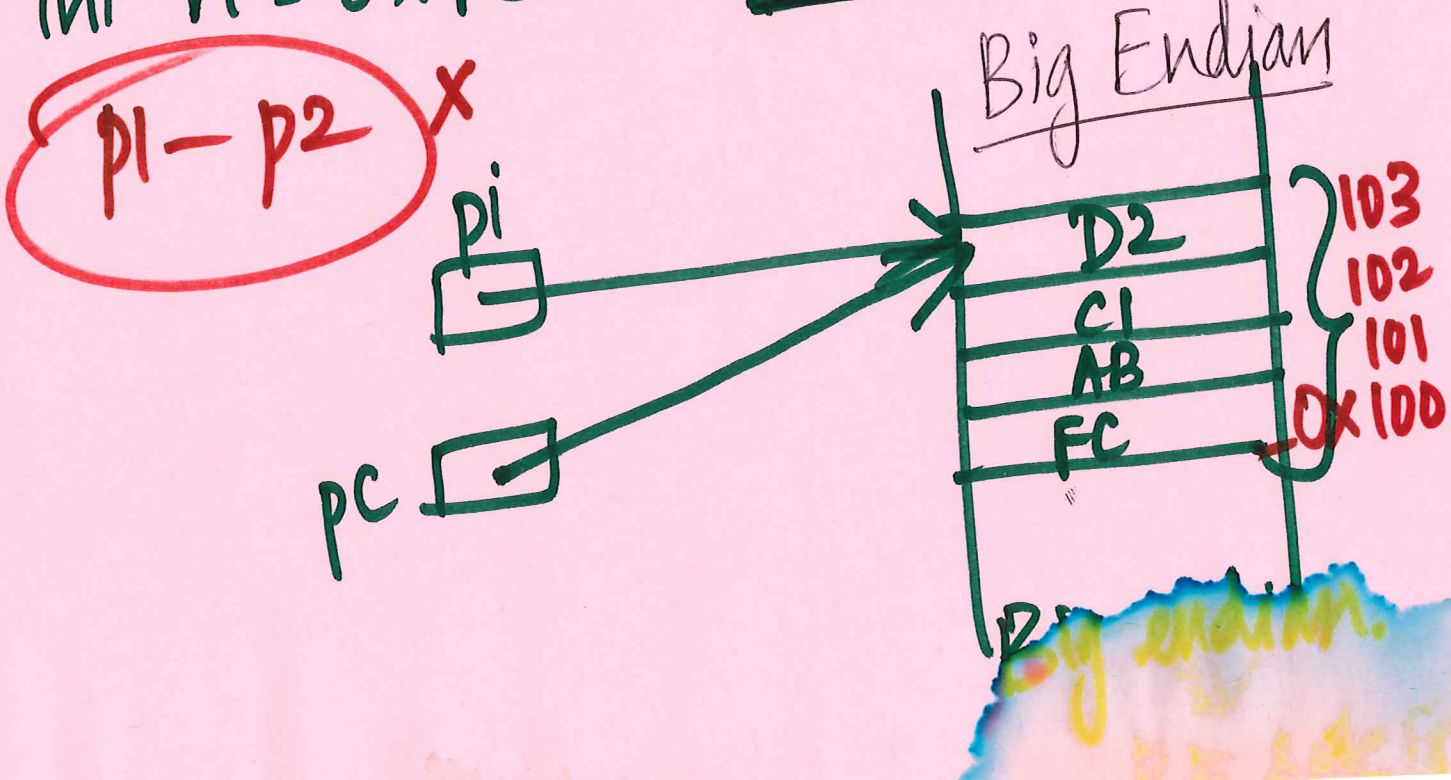
int n= p2 − p1;

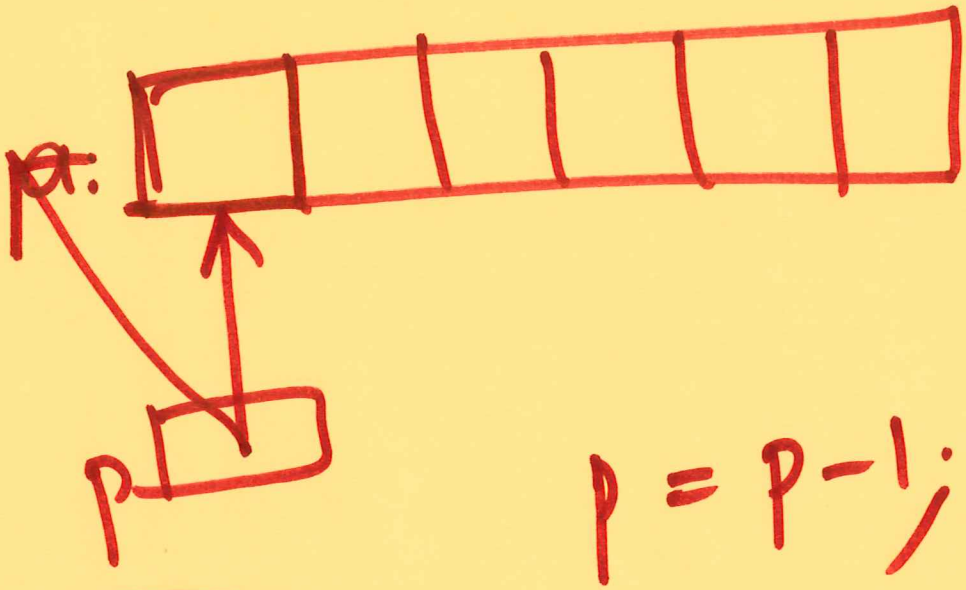
5

28 − 8 = 20

int n = 0x FC AB C1 D2 A

p1 − p2 ✗





p = p - 1;

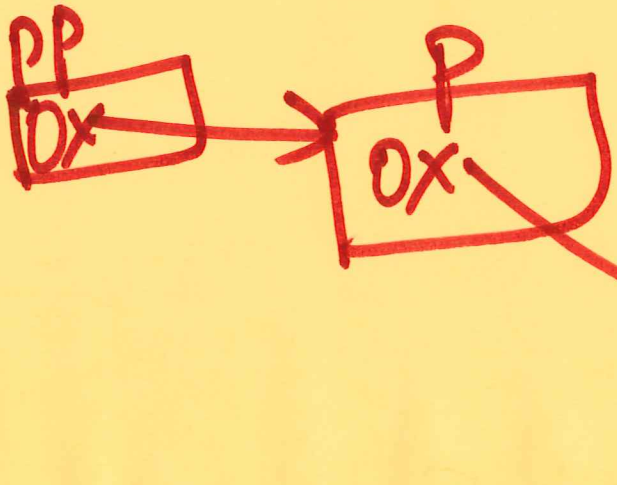
p1 - p2

Not sure.

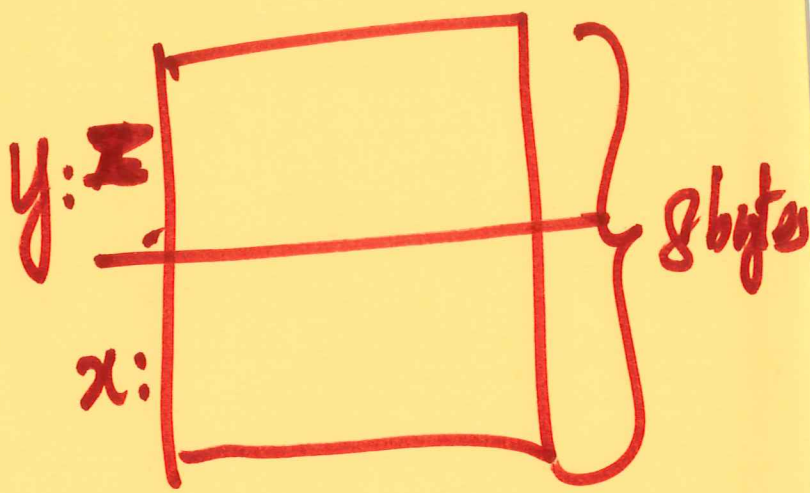
a[-1] ?

int **pp;

Little Endian

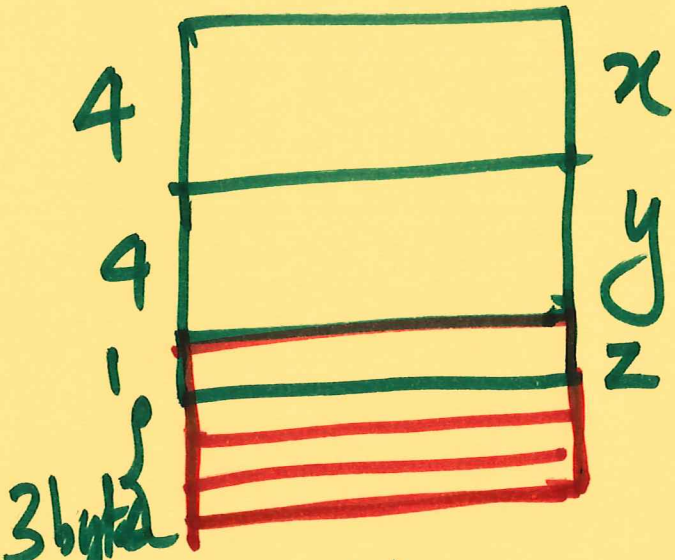


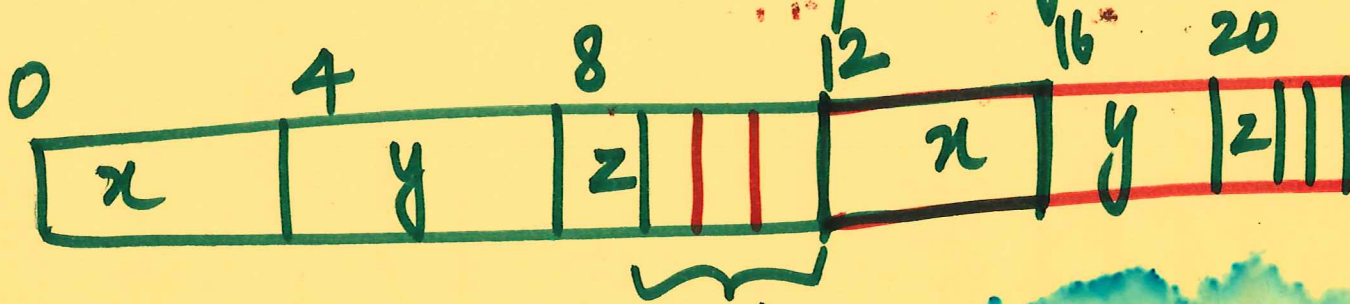
```
struct point
{
    int x;
    int y;
};
```



---

①
```
struct
{
    int x;
    int y;
    char z;
};
```



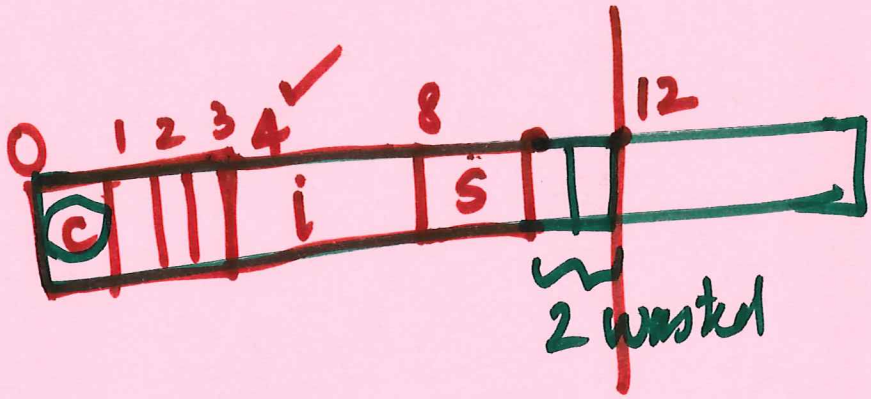


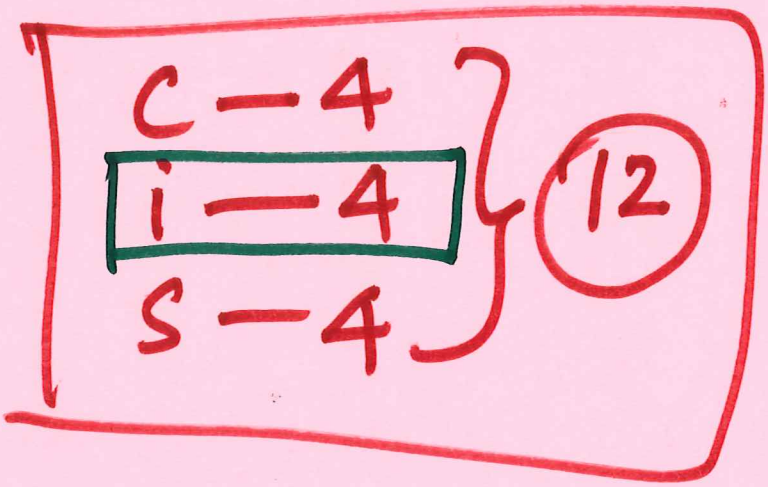
short — memory % by 2

int — " " " 4

char — " " " 1

```
char c;
int i;
short s;
```



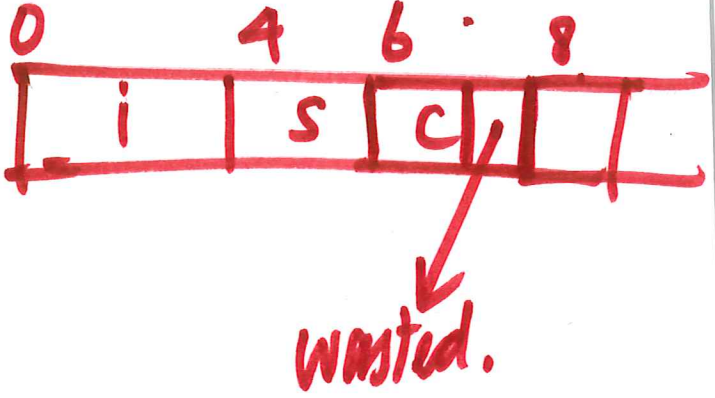


4 , 16

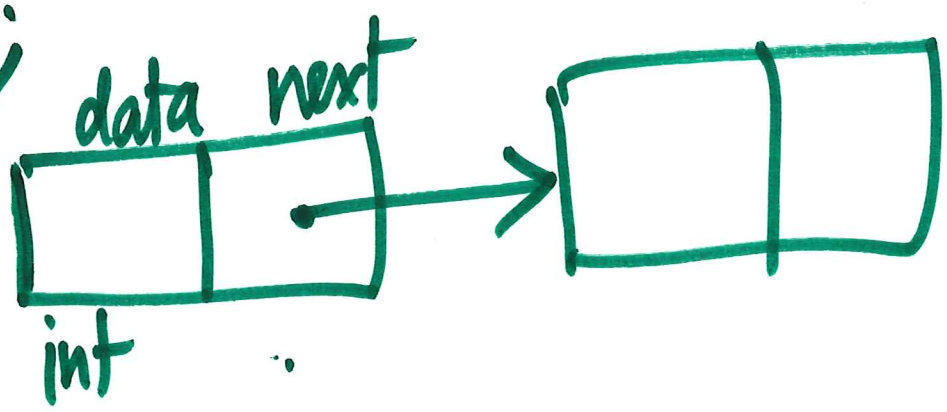
```
struct point pt[10];
```

12 × 10 = 120

int i; → 4
short s; → 2
char c; → 1
waste → 1




```
struct node
{  int data;
     struct node *next;
};
```



&	and
|	or
~	not
^	xor

&&	log. and
||	log. or
!	log NOT.

X = 1011

Y = 0101

X & y = 0001

X && y = 1

1 1

2's complement

4-bit number

1	0	1	1
-8	4	2	1

sign bit

$$\underset{-8}{1}\ \underset{4}{0}\ \underset{2}{1}\ \underset{1}{1} = -8+0+2+1$$

$$= -5$$

$= +7$

0 1 0 1

↓ flip

1 0 1 0

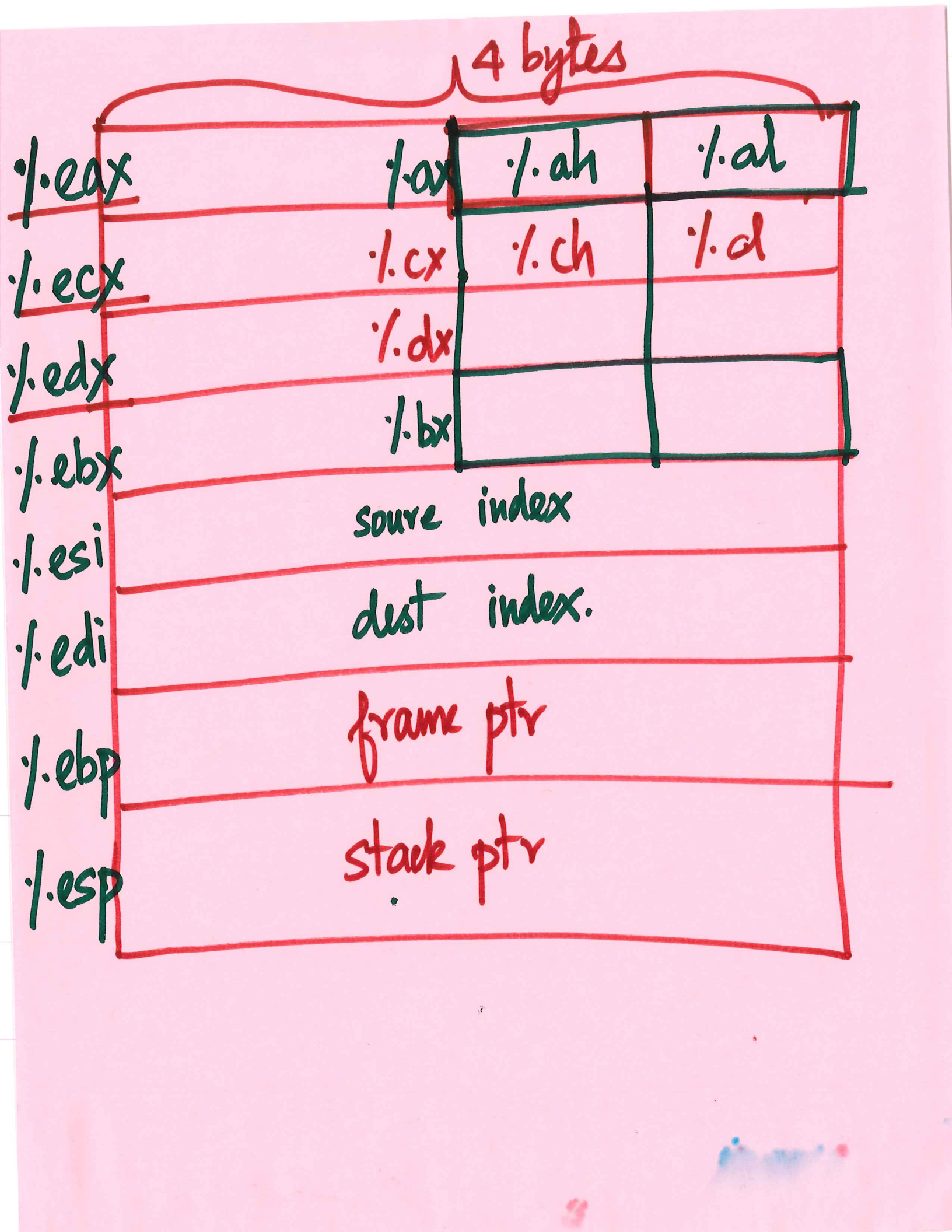
↓ add 1

1 0 1 1 $= -5$

int → signed (2's comp)

unsigned int

$= 8+2+1 = 11$


4 bytes
%eax
%ax
%ah
%al
%ecx
%cx
%ch
%cl
%edx
%dx
%ebx
%bx
%esi
soure index
%edi
dest index.
%ebp
frame ptr
%esp
stack ptr

mov s, d

Assembly - AT&T

1) mov $100, %.eax → imm

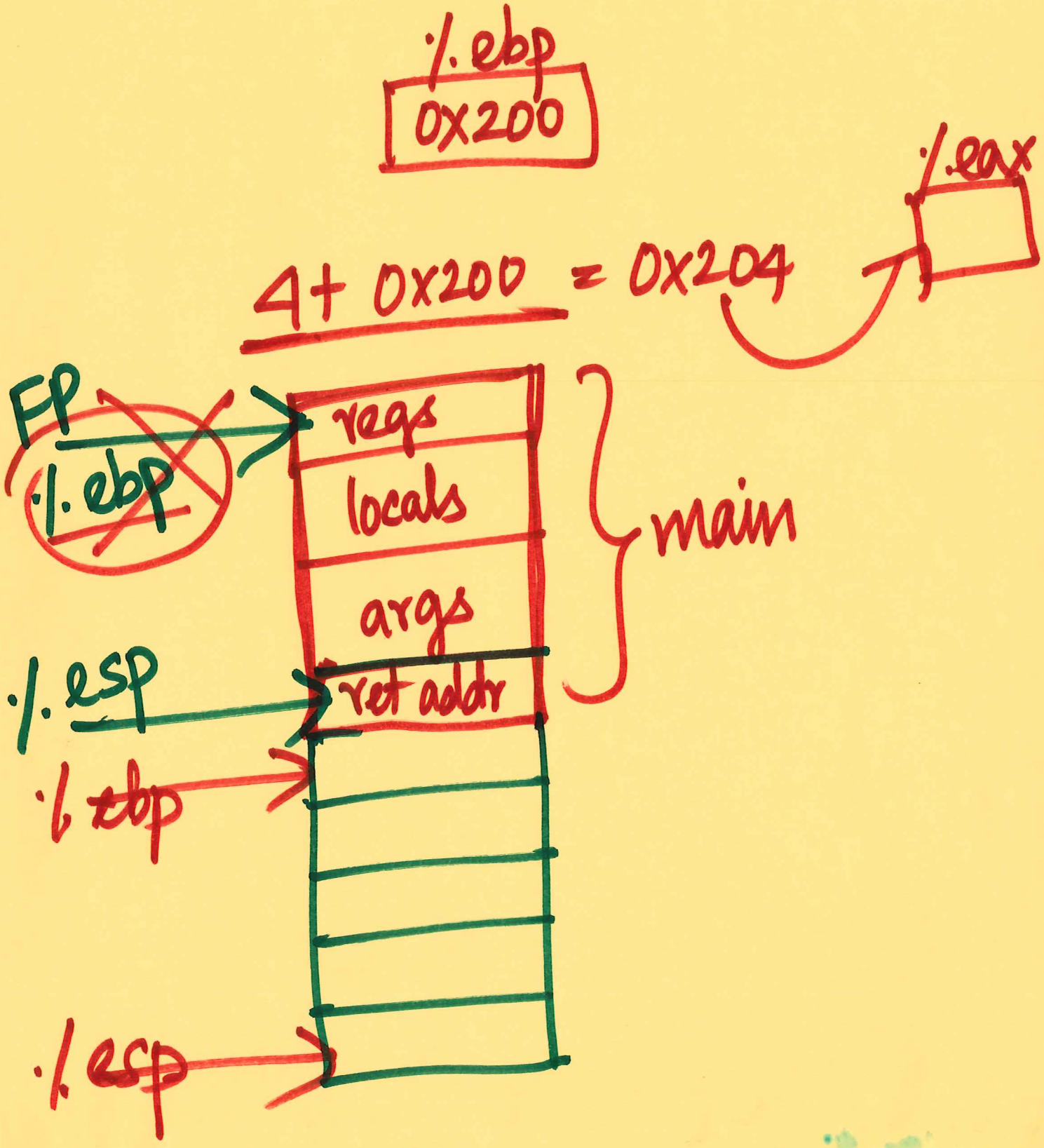
2) mov %.edx, %.eax → reg

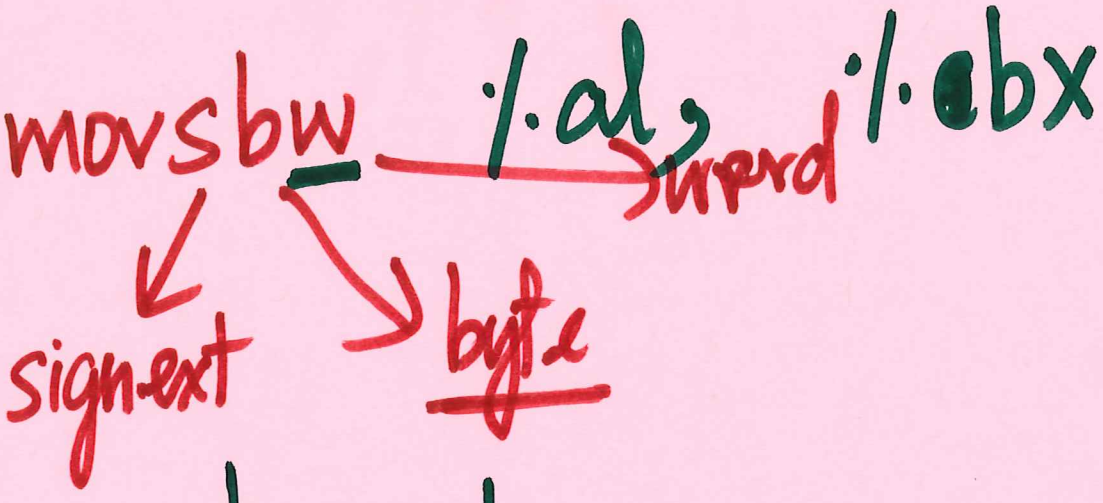
3) mov 0x100, %.eax → mem.

4) mov 8(%.ebx, %.esi, 4), %.eax

M[8 + %.ebx + %.esi * 4]

leal 4(%ebp), %eax





movzbw

movsbl

movswl

addl $1, %eax

⇓

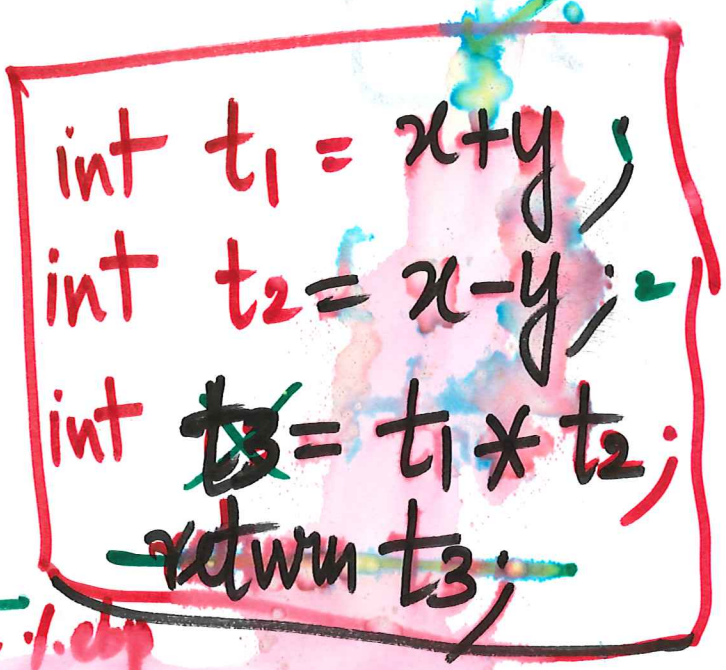
%eax = %eax + 1

```
        _____
    else
        _____
  }
  return sumA;
}
```

10. Write the C code for the following assembly language code.

Note: The C code can be written using less than 5 lines of code.

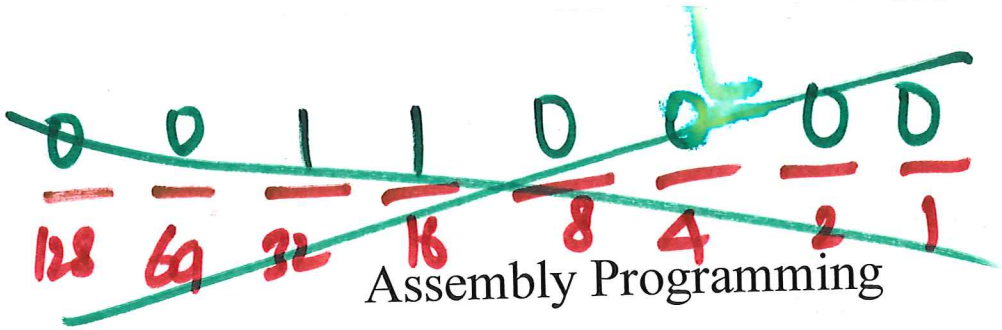
```
mystery:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    8(%ebp), %edx
    movl    12(%ebp), %eax
    addl    %edx, %eax
    movl    %eax, -4(%ebp)
    movl    8(%ebp), %eax
    subl    12(%ebp), %eax
    movl    %eax, -8(%ebp)
    movl    -4(%ebp), %eax
    imull   -8(%ebp), %eax
    movl    %eax, -12(%ebp)
    movl    -12(%ebp), %eax
    leave
    ret

int mystery(int x, int y)
{




}
```



8+%ebp

12+%ebp

return t1 * t2;



Assembly Programming

Note: 3 more practice questions in Assembly were already discussed in class on Friday, March 11th 2016!

8. Assembly to C

char someFunc(int a, int b, int c);

The arguments for this function are on the stack at the following addresses:

int a ⇒ `0x8(%ebp)`
int b ⇒ `0xc(%ebp)`
int c ⇒ `0x10(%ebp)`

Complete the following line with a valid C expression.

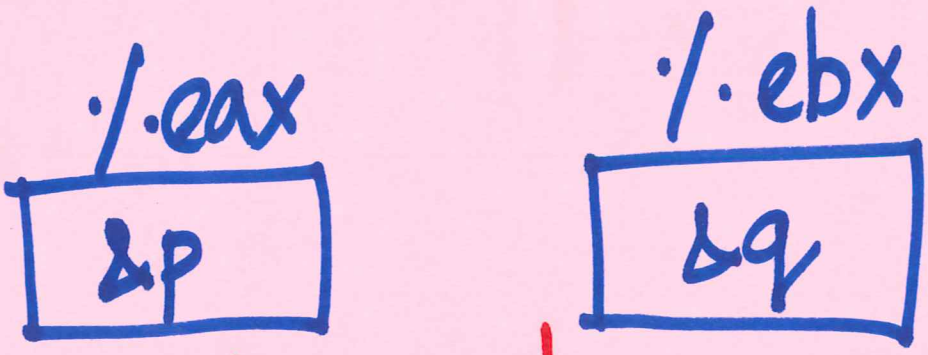
char t5 = `(a-b)>0;`

```
4e: 8b 45 0c      mov    0xc(%ebp),%eax
51: 8b 55 08      mov    0x8(%ebp),%edx
54: 89 d1         mov    %edx,%ecx
56: 29 c1         sub    %eax,%ecx
58: 89 c8         mov    %ecx,%eax
5a: 85 c0         test   %eax,%eax
5c: 0f 9f c0      setg   %al
5f: 88 45 fe      mov    %al,-0x2(%ebp)
```

9. Consider the following assembly code for a C function named loops( ). The line numbers are given in decimal.

```
char loops(int a, int b);

1:          push   %ebp
2:          mov    %esp,%ebp
3:          sub    $0x10,%esp

            //int sumA = a;
4:          mov    0x8(%ebp),%eax
```

1.



s d

AT&T syntax

1. movl (%eax), %esi
2. movl (%esi), %edi
3. movl %edi, (%ebx)

---

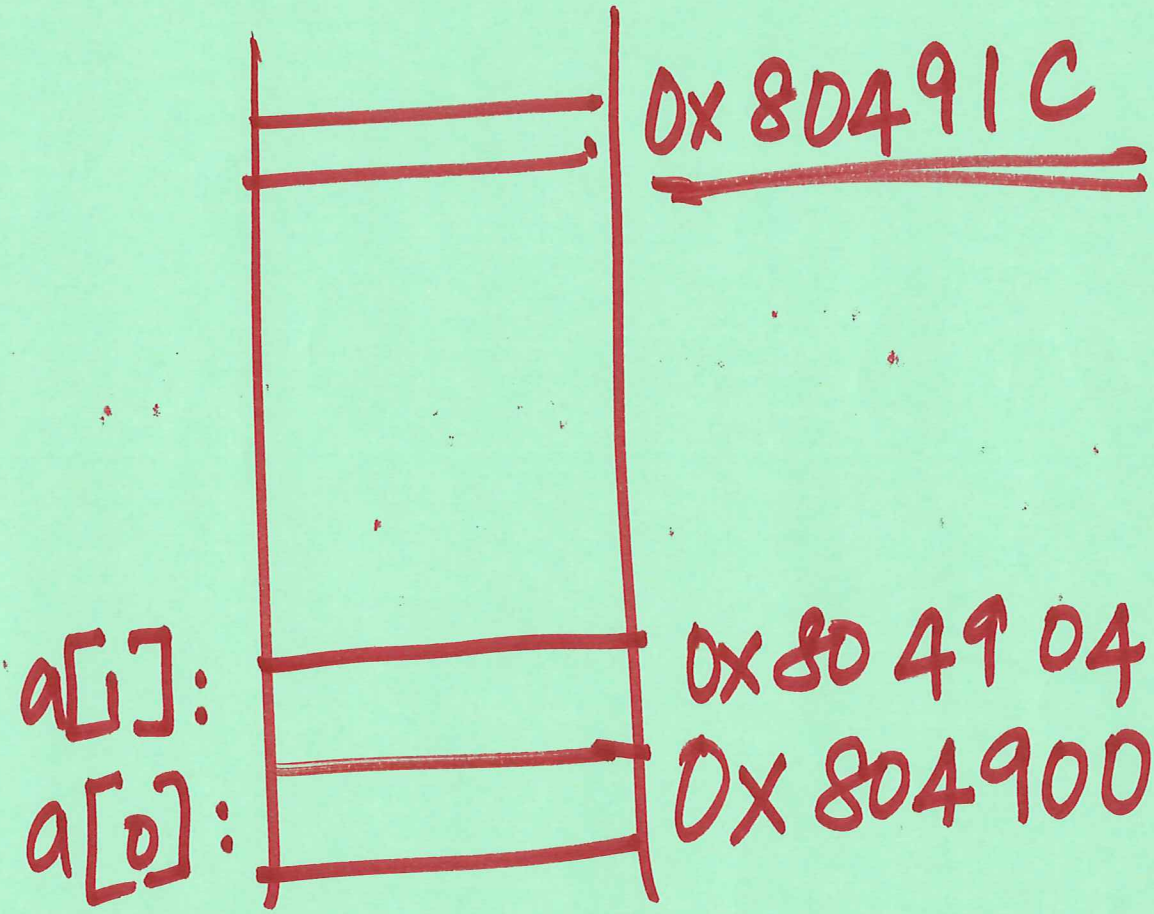
1. *(&p) = p

2. *p ⟶ %edi *p

3. *(&q) = q (dest) ⟹ q = *p;

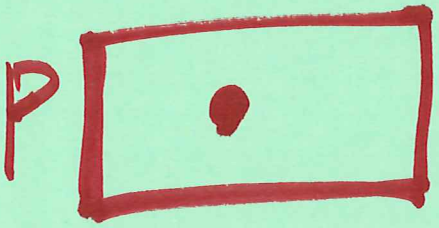
2. int a[10];

int *p = &a[7];



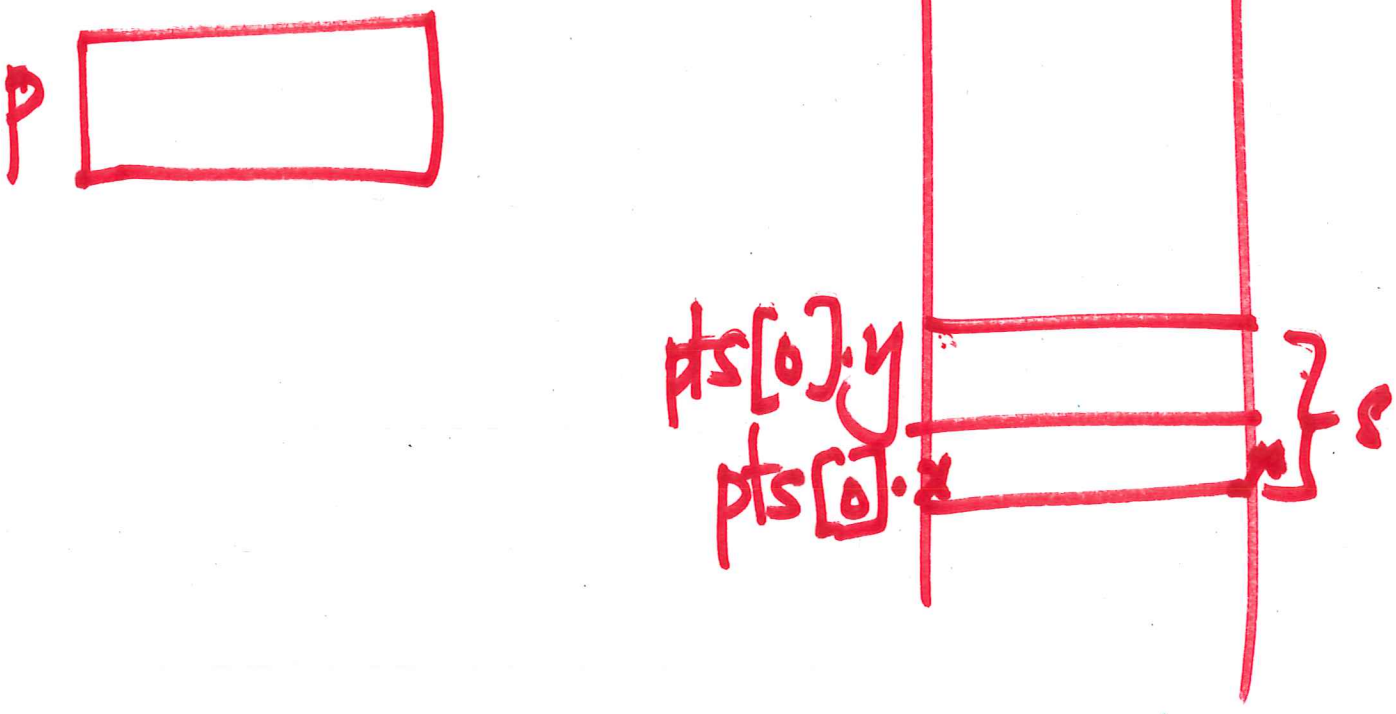
4 x 7 = 28.

= 0x1C



&a[7] = 0x 804900
+ 0x 1C

```
struct point {
    int x;
    int y;
};
struct point pts[10];
int *p = &(pts[3].y);
```



Type casting - K&R C.

int n = 0x AB 01 CC

short s = n;

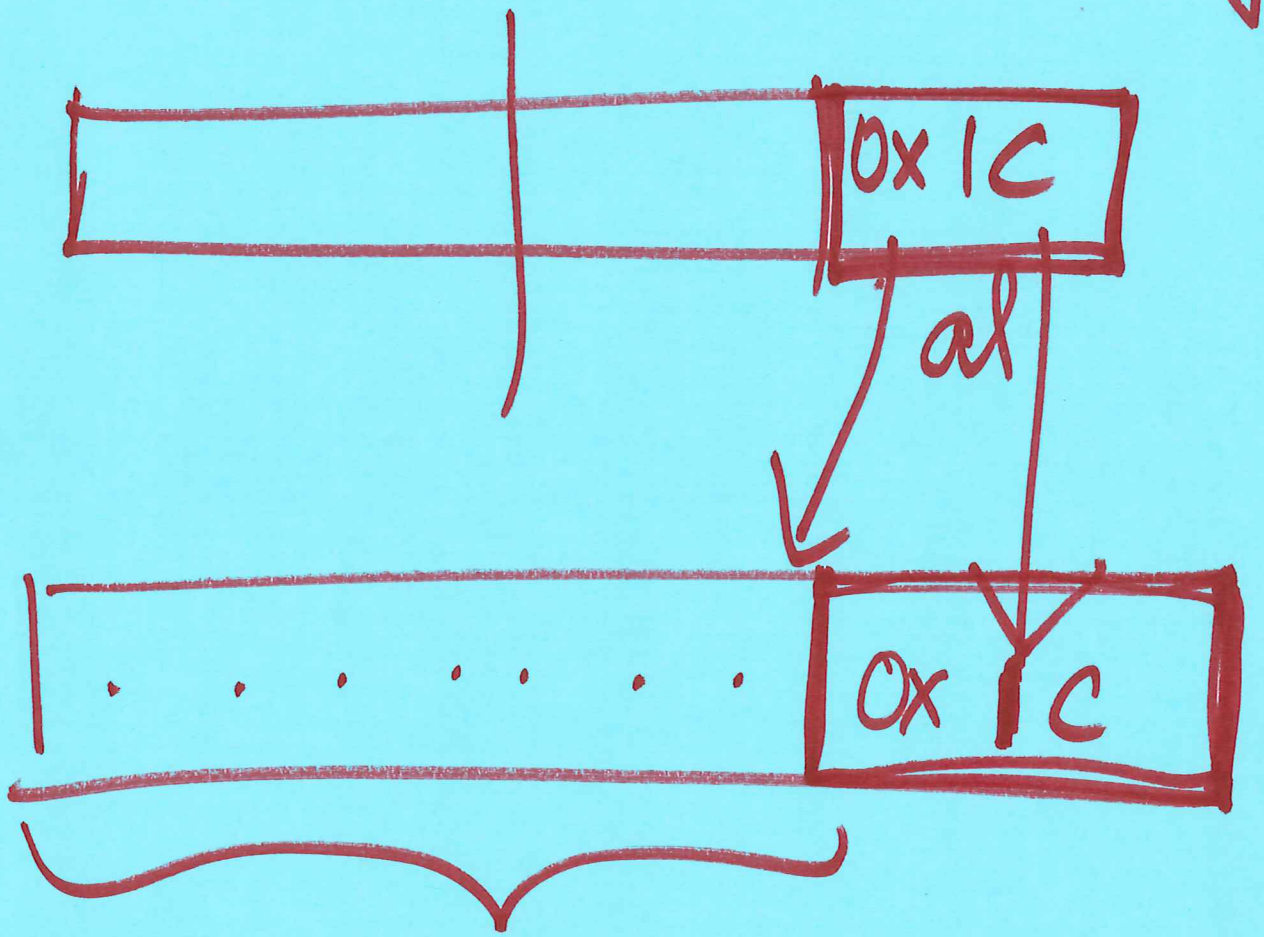
~~int~~ n = s; ⊗ 01 CC

n: 01 CC

short s = (short) n;

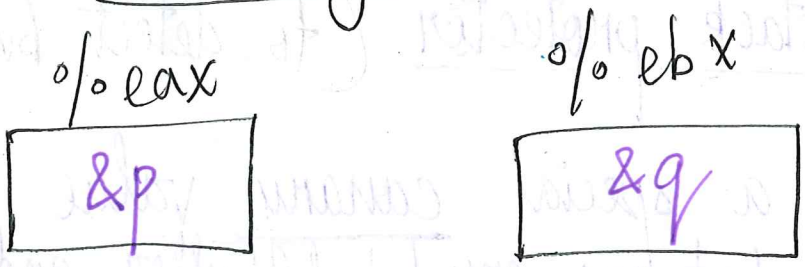
I Know what I'm doing!

char t = (char.) x ____ (unsigned) y;



④ Assembly Review

1.

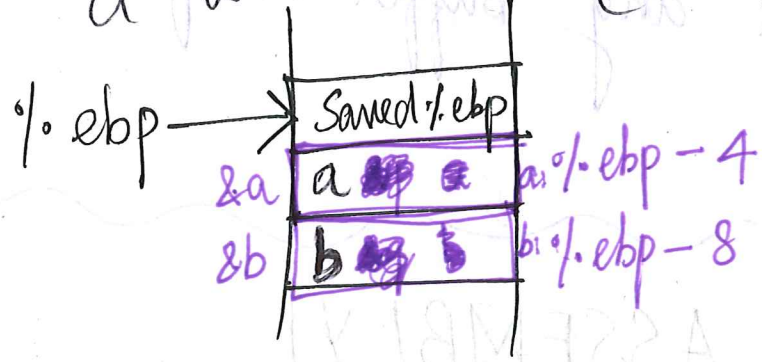


movl (%eax), %esi ⇒ %esi: p

movl (%esi), %edi ⇒ %edi: *p

movl %edi, (%ebx) ⇒ q = *p;

a is in -0x4(%ebp) | b is in -0x8(%ebp)

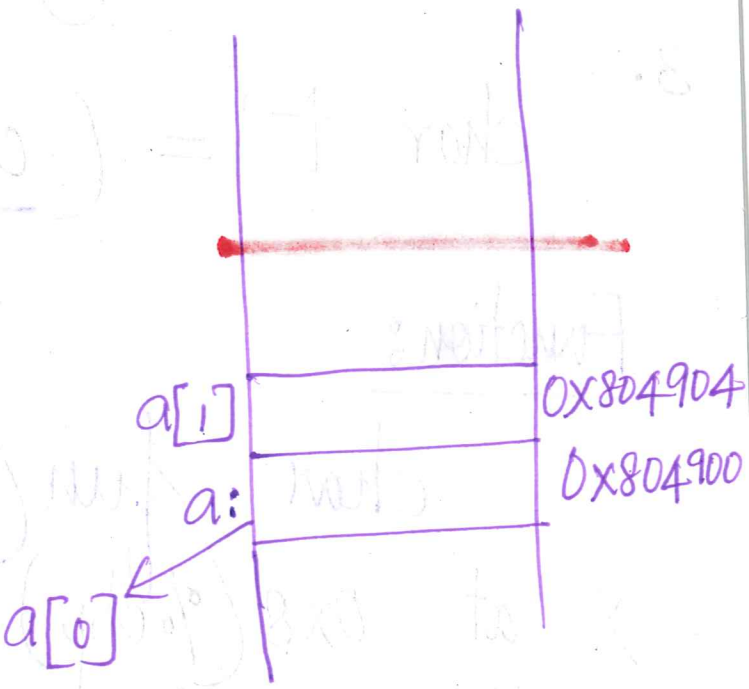


b = *q;

- movl -0x4(%ebp), %eax
- movl (%eax), %eax
- movl %eax, -0x8(%ebp)

⇒ ~~q = *p;~~

2. `int a[10];`
`int *p = &a[7];`

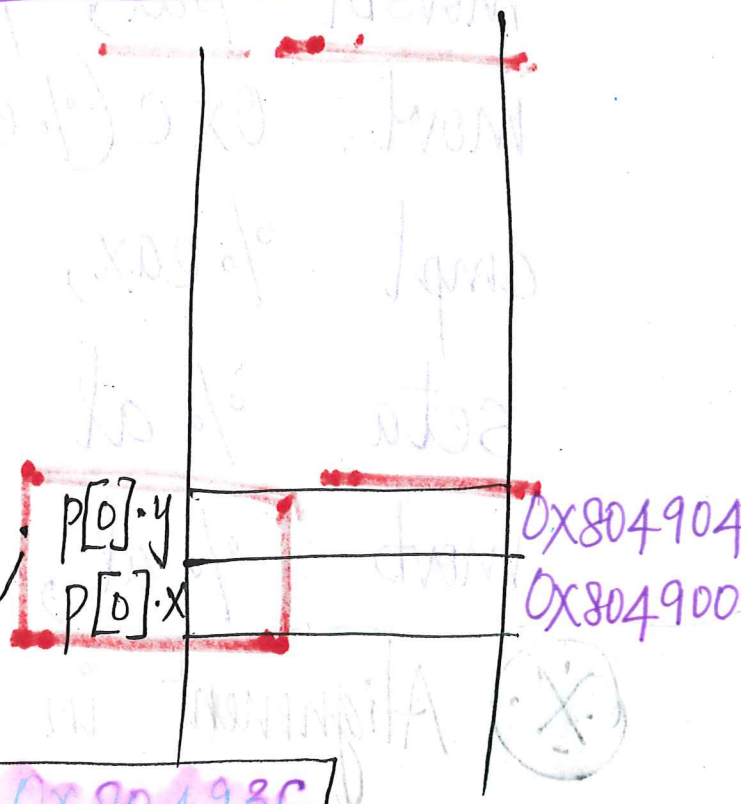


p = ?

7 X 4 = 28 = 0x1C

∴ a[7] is at 0x 804900
+ 0x 1C
0x 80491C

```
struct point {
    int x;
    int y;
};
struct point p[10];
int *ptr = &(p[7].y);
```



8 X 7 = 56 = 0x38

ptr = 0x 804900
+ 0x 38
→ ptr = 0x 80493C