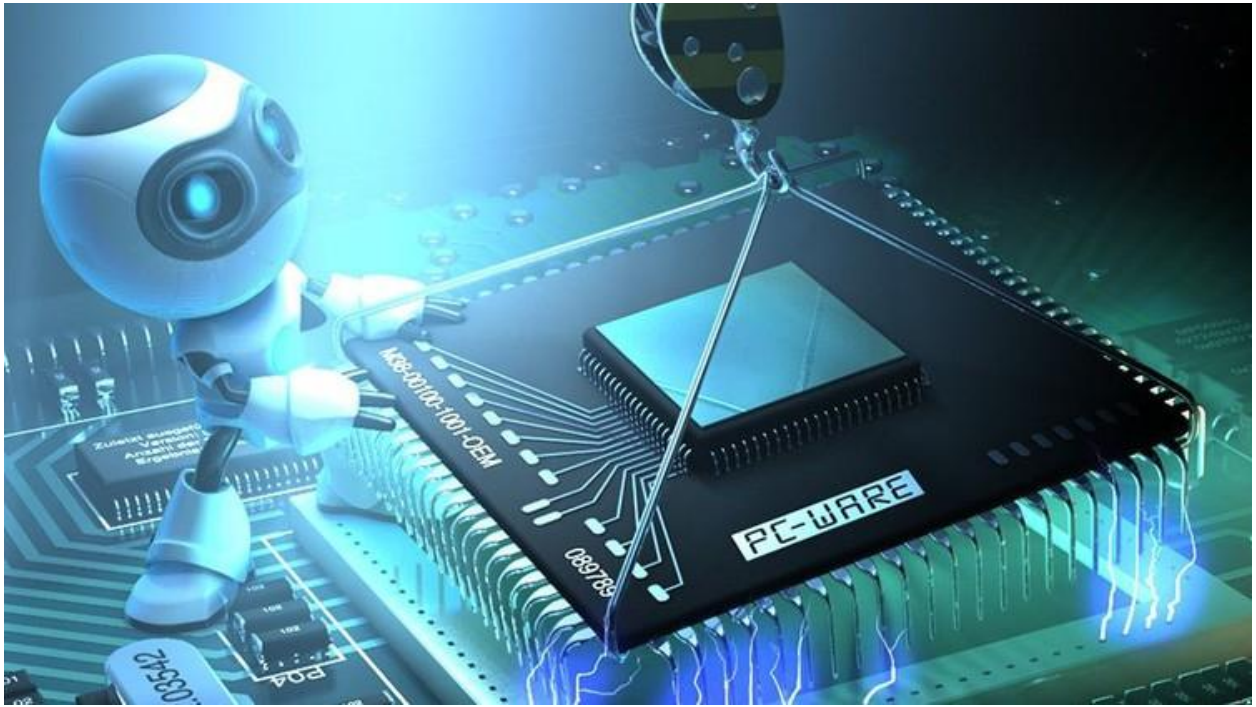
Exam 5 - Problem 1



Exam 5 - Problem 1

Given this code in C:

```
typedef struct {
    char c;
    short s[4];
    char c2;
    int i2;
} sx;
int  exam  (sx *p1, sx s1, short sh){
    int i;
    char ch;
    short aux;
    int *x;
    . . .
}
```

Exam 5 - Problem 1

a) Draw how the `sx` structure would be stored, clearly indicating the offsets from the start and the size of all the fields.

b) Draw the activation block of the exam routine, clearly indicating the offsets with respect to %ebp and the size of all fields.

c) Translate to x86 assembler the statement `return(*x + s1.i2);` assuming you are inside the exam function:

d) Translate to x86 assembler the statement `(*p1).s[2] = aux;` assuming it is inside the exam function.

Exam 5 - Problem 1

a) Draw how the `sx` structure would be stored, clearly indicating the offsets from the start and the size of all the fields.

```
typedef struct {
    char c;
    short s[4];
    char c2;
    int i2;
} sx;
```

Exam 5 - Problem 1

a) Draw how the `sx` structure would be stored, clearly indicating the offsets from the start and the size of all the fields.

```
typedef struct {
    char c;
    short s[4];
    char c2;
    int i2;
} sx;
```

```
---------------------
|  s[0]   |  -- | c | c <- +0; s[0]<- +2
---------------------
|  s[2]   |  s[1]   | s[1]<- +4; s[2]<- +6;
---------------------
| --  c2  |  s[3]   | s[3]<- +8; c2 <-10
---------------------
|       i2          | i2<- +12 // size 16 bytes
---------------------
```

Exam 5 - Problem 1

b) Draw the activation block of the exam routine, clearly indicating the offsets with respect to %ebp and the size of all fields.

```
int exam (sx *p1, sx s1, short sh){
    int i;
    char ch;
    short aux;
    int *x;
    . . .
}
```

Exam 5 - Problem 1

b) Draw the activation block of the exam routine, clearly indicating the offsets with respect to %ebp and the size of all fields.

```
--------------------
|         i         | i<- ebp-12;
--------------------
| aux     | --  ch  | ch <- ebp-8 ; aux<- ebp-6
--------------------
|       *x          | *x <- ebp-4
--------------------
|        ebp        | <- ebp
--------------------
|        RET        |
--------------------
|        *p1        | p1<- ebp+8
--------------------
|  s[0]   |  -- | c |  c<- ebp+12; s[0]<- ebp+14
--------------------
|  s[2]   |  s[1]   | s[1]<- ebp+16; s[2]<- ebp+18;
--------------------
| --  c2  |  s[3]   | s[3]<- ebp+20; c2<- ebp+22
--------------------
|        i2         | i2<- ebp+24
--------------------
| --   -- |   sh    | sh<- ebp +28
--------------------
```

```
int exam (sx *p1, sx s1, short sh){
   int i;
   char ch;
   short aux;
   int *x;
   . . .
}
```

```
typedef struct {
   char c;
   short s[4];
   char c2;
   int i2;
} sx;
```

Size: 44 bytes

Exam 5 - Problem 1

c) Translate to x86 assembler the statement `return(*x + s1.i2);` assuming you are inside the exam function:

Exam 5 - Problem 1

c) Translate to x86 assembler the statement `return(*x + s1.i2);` assuming you are inside the exam function:

```
movl -4(%ebp),%ecx   # %ecx = x
movl (%ecx), %eax    # %eax = *x
addl 24(%ebp),%eax   # %eax = *x + s1.i2
movl %ebp, %esp
popl %ebp
ret
```

Exam 5 - Problem 1

c) Translate to x86 assembler the statement `return(*x + s1.i2);` assuming you are inside the exam function:

```
movl -4(%ebp),%ecx   # %ecx = x
movl (%ecx), %eax    # %eax = *x
addl 24(%ebp),%eax   # %eax = *x + s1.i2
movl %ebp, %esp
popl %ebp
ret
```

Very important to remember to save the result inside %eax

Exam 5 - Problem 1

d) Translate to x86 assembler the statement `(*p1).s[2] = aux;` assuming it is inside the exam function.

Exam 5 - Problem 1

d) Translate to x86 assembler the statement `(*p1).s[2] = aux;` assuming it is inside the exam function.

```
movl 8(%ebp),%ecx      # %ecx = &p1
movw -6(%ebp),%ax      # %ax = aux
movw %ax,6(%ecx)       # (*p1).s[2] = aux
                       # Here 6 (%ecx) is equivalent to (*p1).s[2]
```