Zad 3.

środa, 1 marca 2023 21:52

Zadanie 3. Podaj rozmiar w bajtach poniższych struktur przyjmując, że wskaźnik jest 64-bitowy (architektura x86-64). Pod jakim przesunięciem, względem początku struktury, znajdują się poszczególne pola? Jak zreorganizować pola struktury, by zajmowała mniej miejsca? Z czego wynika takie zachowanie kompilatora?

```
1 struct A {
2   int8_t a;
3   void *b;
4   int8_t c;
5   int16_t d;
6 };
```

```
1 struct B {
2   uint16_t a;
3   double b;
4   void *c;
5 };
```

Wskazówka: Użyj kompilatora, aby się dowiedzieć jaki jest rozmiar powyższych struktur – przyda się słowo kluczowe «sizeof».

Data structure padding [edit]

Although the compiler (or interpreter) normally allocates individual data items on aligned boundaries, data structures often have members with different alignment requirements. To maintain proper alignment the translator normally inserts additional unnamed data members so that each member is properly aligned. In addition, the data structure as a whole may be padded with a final unnamed member. This allows each member of an array of structures to be properly aligned.

całość ma być wielokrotnością największego membera

Padding is only inserted when a structure member is followed by a member with a larger alignment requirement or at the end of the structure. By changing the ordering of members in a structure, it is possible to change the amount of padding required to maintain alignment.

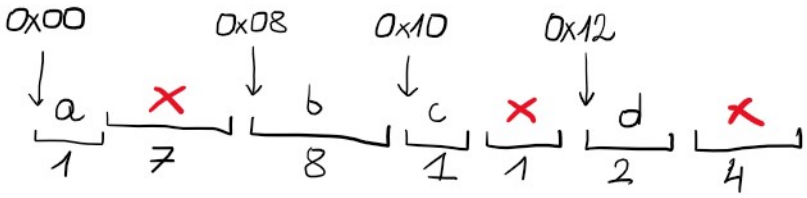
C Data Type	Typical 32-bit	Typical 64-bit	x86-64
char	1	1	1
short	2	2	2
int	4	4	4
long	4	8	8
float	4	4	4
double	8	8	8
pointer	4	8	8

```
1 struct A {
2   int8_t a;
3   void *b;
4   int8_t c;
5   int16_t d;
6 };
```

1
8
1
2

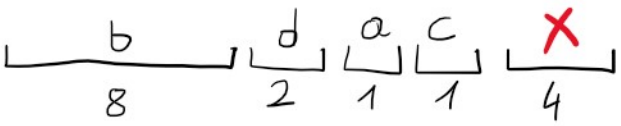
sizeof(A) = 24

member	offset
a	0x00
b	0x08
c	0x10
d	0x12



lepszy będzie taki

```
1 struct A {
2    void *b;
3    int16_t d;
4    int8_t a;
5   int8_t c;
6 };
```



sizeof(A) = 16

```
1 struct B {
2    uint16_t a; 2
3    double b; 8
4    void *c;  8
5 };
```

member	offset
a	0x00
b	0x08
c	0x10

sizeof(B) = 24

nie da się poprawić

2+8+8 = 18 ⇒ zawsze conajmniej 24, bo ma być wielokrotnością 8

The CPU accesses memory by a single memory word at a time. As long as the memory word size is at least as large as the largest primitive data type supported by the computer, aligned accesses will always access a single memory word. This may not be true for misaligned data accesses.

If the highest and lowest bytes in a datum are not within the same memory word the computer must split the datum access into multiple memory accesses. This requires a lot of complex circuitry to generate the memory accesses and coordinate them. [redacted]