Lista 7

tags: ASK

Zadanie 1

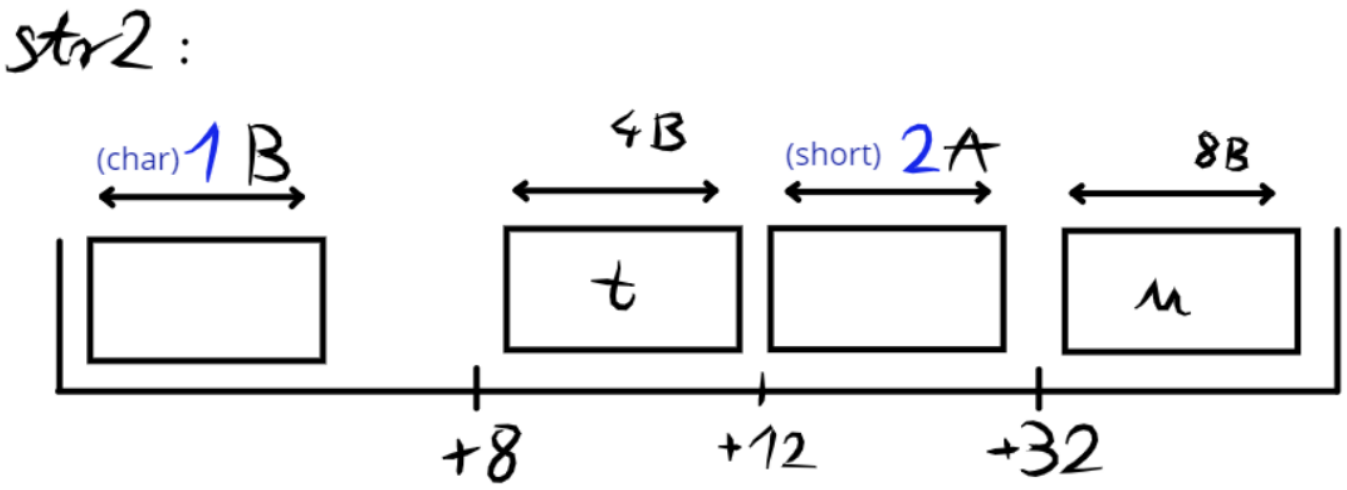
Zadanie 1. Przeczytaj poniższy kod w języku C i odpowiadający mu kod w asemblerze, a następnie wywnioskuj jakie są wartości stałych «A» i «B».

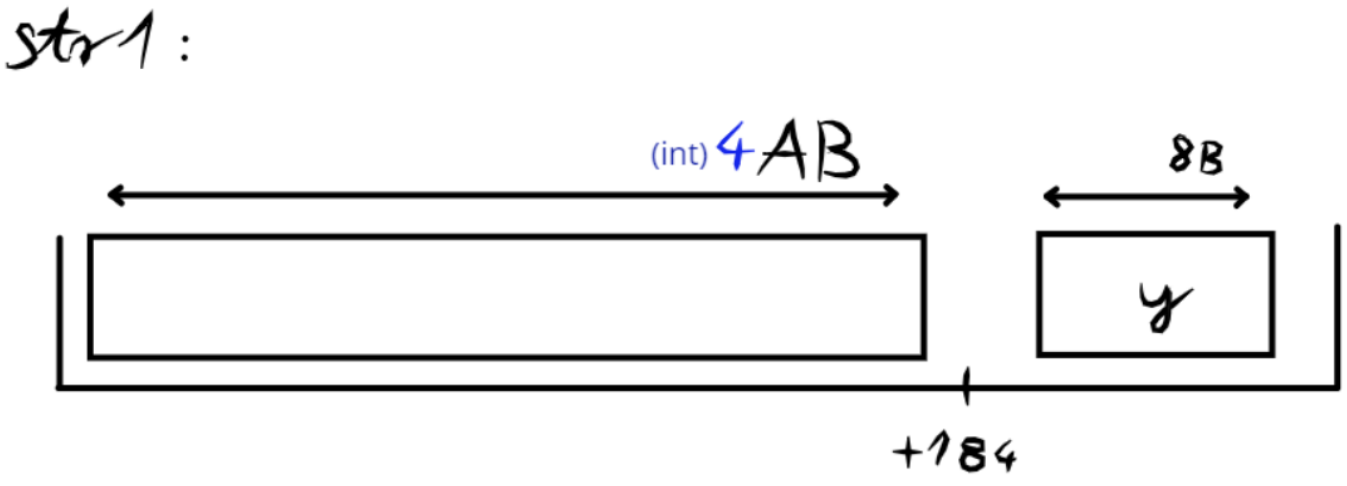
```
typedef struct {
    int x[A][B];   //4B
    long y;        //8B
} str1;

typedef struct {
    char array[B]; //1B
    int t;         //4B
    short s[A];    //2B
    long u;        //8B
} str2;

void set_val(str1 *p, str2 *q) {
    long v1 = q->t;
    long v2 = q->u;
    p->y = v1 + v2;
}
```

```
set_val:
    movslq 8(%rsi),%rax
    addq 32(%rsi),%rax
    movq %rax,184(%rdi)
    ret
```





Obserwacje:

- $4 < B \le 8$ (gdyby tak nie było to t byłoby wyrównane i zaczynałoby się na +4)
- $12 < 2A \le 20 => 6 < A \le 10$ (ten sam powód co powyżej, ale wyrównanie jest dla 8 bajtów)
- $176 < 4AB \le 184 => AB \in {45, 46}$

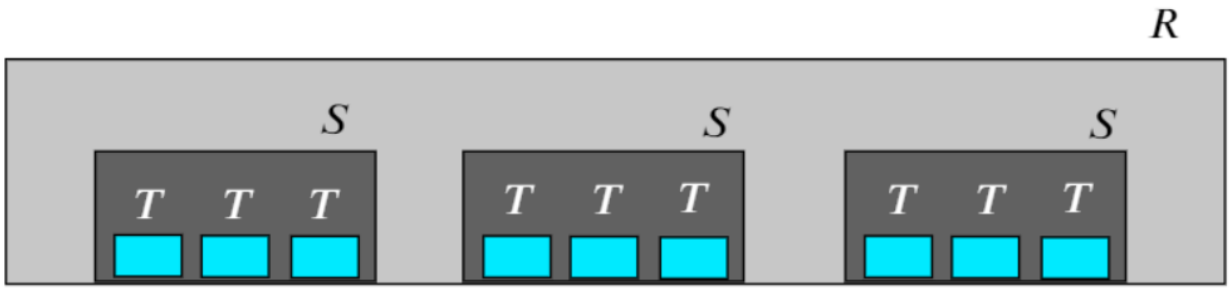
Jedynym rozwiązaniem takie układu równań jest $A = 9, B = 5$

Zadanie 2

Zadanie 2. Przeczytaj poniższy kod w języku C i odpowiadający mu kod w asemblerze, a następnie wywnioskuj jakie są wartości stałych «R», «S» i «T».

```
long A[R][S][T];
long store_elem(long i, long j, long k, long *dest)
{
    *dest = A[i][j][k];
    return sizeof(A);
}
```

```
store_elem:
    leaq (%rsi,%rsi,2),%rax //rax = 3j
    leaq (%rsi,%rax,4),%rax //rax = 13j
    movq %rdi,%rsi          //rsi = i
    salq $6,%rsi            //rsi = 64i
    addq %rsi,%rdi          //rdi = 65i
    addq %rax,%rdi          //rdi = 13j + 65i
    addq %rdi,%rdx          //rdx = 13j + 65i + k
    movq A(,%rdx,8),%rax    //rax = *(A + 8(65i + 13j + k))
    movq %rax,(%rcx)        //dest = A[i][j][k]
    movq $3640,%rax         //sizeof(A) = 8 * R * S * T = 3640
    ret
```



Widzimy, że aby dostać się do elementu tablicy R o indeksie i musimy przeskoczyć $65i \cdot 8$ bajtów. Dla elementów tablicy S mamy skok $13j \cdot 8$ bajtów. Stąd wynika, że podtablice T są 13-elementowe, czyli ich rozmiar to $13 \cdot 8$ bajtów. Wtedy $S = 65 / 13 = 5$ oraz $R = 3640/(65 \cdot 8) = 7$.

Zadanie 3

Zadanie 3. Przeczytaj poniższy kod w języku C i odpowiadający mu kod w asemblerze, a następnie wywnioskuj jaka jest wartość stałej «CNT» i jak wygląda definicja struktury «a_struct».

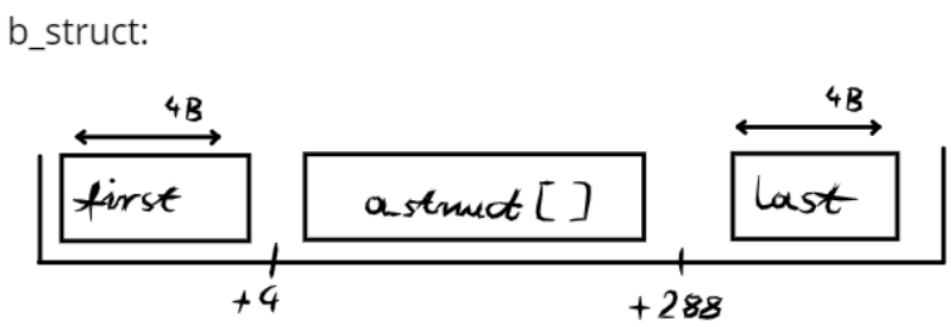
```
typedef struct {
    int first;
    a_struct a[CNT];
    int last;
} b_struct;

void test (long i, b_struct *bp) {
    int n = bp->first + bp->last;
    a_struct *ap = &bp->a[i];
    ap->x[ap->idx] = n;
}
```

```
test:
    movl 0x120(%rsi),%ecx         # ecx = bp->last
    addl (%rsi),%ecx              # ecx = bp->first + bp->last
    leaq (%rdi,%rdi,4),%rax       # rax = 5i
    leaq (%rsi,%rax,8),%rax       # rax = 40i + bp
                                  #
    movq 0x8(%rax),%rdx           # idx = bp + 8 + 40i
                                  # wchodzimy do bp, przeskakujemy int first i
                                  # wybieramy i-ty element tablicy `a`
                                  # i wyciągamy siedzącą tam wartość do rdx (=idx)
    movslq %ecx,%rcx              # sign extend long to quad n
    movq %rcx,0x10(%rax,%rdx,8)   # ładujemy n do (rax + (8*idx) + 16) (= bp + 8 + 40i + 8 + (8*idx))
                                  # czyli do x[idx]
    retq
```

Obserwacje:

- $sizeof(a_struct)$ to 40B
- elementy tablicy x są rozmiaru 8B
- idx to $long$



Jako, że tablica a znajmuje co najwyżej 284 bajty mamy $CNT = 284 / 40 = 7$ (tak naprawdę to 7.1 nie odejmując paddingu).

```
typedef struct {
    long idx;
    long x[4]; //[4] ponieważ rozmiar a_struct to 40B (5*8)
} a_struct;
```

Zadanie 4

Zadanie 4. Przeczytaj definicję unii «elem» i wyznacz jej rozmiar w bajtach. Następnie przepisz procedurę «proc» na kod w języku C.

```
union elem {
    struct {
        long *p;
        long y;
    } e1;
    struct {
        long x;
        union elem *next;
    } e2;
};
```

%rdi jest wskaźnikiem ptr na structa

```
proc:
    movq 8(%rdi),%rax    # rax = *(ptr + 8)  # ptr->e2.next (wynika z następnych dwóch wierszy)
    movq (%rax),%rdx     # rdx = *rax        # next->e1.p (dereferujemy wskaźnik next)
    movq (%rdx),%rdx     # rdx = *rdx        # rdx = *p
    subq 8(%rax),%rdx    # rdx -= *(rax + 8) # rdx = *p - y
    movq %rdx,(%rdi)     # *rdi = rdx        # x = *p - y
                                             # (wiemy, z pierwszego wiersza, że rdi to e2)
    ret
```

```
union elem* proc(union elem *ptr){
    union elem *next = ptr -> e2.next;
    long p_value = *(next -> e1.p);
    ptr -> e2.x = p_value - next -> e1.y;
    return next;
}
```

Zadanie 6

Zadanie 6. Poniżej widniej kod procedury o sygnaturze «float puzzle6(struct P *, float)». Wyznacz definicję typu «struct P». Przetłumacz tę procedurę na język C i wyjaśnij jednym zdaniem co robi.

- $vfmadd231ss$ xmm1, xmm2, xmm3 => xmm3 += xmm1 * xmm2
- $vmulss$ xmm1, xmm2, xmm3 => xmm3 = xmm1 * xmm2

```
puzzle6:
    movq (%rdi), %rdx                                   # pierwsze pole structa (:= n)
    leaq 8(%rdi), %rcx                                  # drugie pole structa (:= x)
    xorl %eax, %eax                                     # rax = 0 (:= i)
    vxorps %xmm1, %xmm1, %xmm1                          # xmm1 = 0.0
    vmovss .LC1(%rip), %xmm2                            # xmm2 = 1.0

#pętla
.L2:
    cmpq %rdx, %rax                                     # jeśli i >= n
    jge .L5                                             # skaczemy do L5
    vfmadd231ss (%rcx,%rax,4), %xmm2, %xmm1             # xmm1 += xmm2 * x[i]
    incq %rax                                           # rax++ (widać, że rax jest tutaj licznikiem)
    vmulss %xmm0, %xmm2, %xmm2                          # xmm2 *= xmm0
    jmp .L2
.L5:
    vmovaps %xmm1, %xmm0                                # zwracamy wynik w %xmm0
    ret

.LC1: .long 0x3f800000 (AKA 1)
```

```
typedef struct {
    long n;
    float x[];
}

float puzzle6(struct P *sp, float p){
    long n = sp->n;
    float sum = 0;
    float power = 1;

    for(int i=0; i<=n; i++){
        sum += ps->x[i] * power;
        power *= p;
    }

    return sum;
}
```

Powyższy kod liczy sumę $\sum_{k=0}^n x_k \cdot p^k$