

Prvi kolokvijum iz Operativnih sistema 1

Odsek za računarsku tehniku i informatiku

April 2024.

1. (10 poena)

a)(6)

```
seg data
    a dd 1, 2, 3, 4, 5
endseg

seg bss
    n dd 0
endseg

seg text
    max:      load r1,[sp+12]    ; r1 := a
              load r2,[sp+16]    ; r2 := n
              load r3,#1         ; if (n<=1)
              cmp  r2,r3
              jg   L0001
              load r0,[r1]       ; return a[0]
              jmp  max_END
    L0001:    sub  r2,r2,r3       ; r1 := n-1
              push r2
              load r4,#4
              add  r5,r1,r4
              push r5
              call max           ; r0 := max(a+1,n-1)
              pop  r5
              pop  r2
              load r1,[sp+12]    ; r1 := a
              load r2,[r1]       ; r2 := a[0]
              cmp  r2,r0         ; (a[0]>m)?
              jle  max_END
              load r0,r2
    max_END:  ret
endseg
```

b)(2) 0xFFFFF3B0 (-20*4 decimalno)

c)(2) Logičke segmente po virtuelnom adresnom prostoru raspoređuje linker. Prilikom prevođenja navedenog poziva, assembler ne može koristiti relativno adresiranje, jer ne zna vrednost adrese odredišta skoka, pa mora koristiti apsolutno (memorijsko direktno) adresiranje. (Teorijski, ovo bi bilo moguće kada bi linker bio sposoban da izračunava konstantne izraze koje ostavlja assembler za nerazrešene vrednosti pomeraja u instrukcijama, pa assembler može da ostavi ovakav izraz u prevedenom fajlu koji potom rešava linker poznajući sve konstante u njemu, ali to nije praktično, jer složenost implementacije ne opravdava dobit.)

2. (10 poena) a(7)

```
inline PageDesc* getPageDesc (PMT0 pmt, void* vaddr) {
    uint32 page = (uint32)vaddr>>offsetw;
    uint32 page0 = page>>pagelw;
    uint32 page1 = page & ~((uint32)-1<<pagelw);
    PMT1* pmt1 = pmt[page0];
    if (!pmt1) pmt1 = (PMT1*)kmalloc(sizeof(PMT1));
    if (!pmt1) return nullptr;
    pmt[page0] = pmt1;
    return &pmt1[page1];
}
```

b(3) Operativni sistem treba najpre da proveri da li tražena virtuelna adresa koja je generisala straničnu grešku pripada nekom alociranom logičkom segmentu virtuelne memorije, na osnovu strukture deskriptora logičkih segmenata (bez pristupa PMT). Ako ne pripada, proces je prestupio u pristupu memorije i operativni sistem može da ga ugasi. Ako pripada, treba da alocira okvir za smeštanje tražene stranice, u zavisnosti od vrste logičkog segmenta učita tu stranicu sa diska, inicijalizuje sadržaj okvira ili ništa od toga (videti objašnjenja na predavanjima), a onda, korišćenjem funkcije realizovane pod a), dohvati deskriptor stranice (i usput alocira PMT drugog nivoa ako je potrebno) i postavi ga na odgovarajuću vrednost (broj okvira i prava pristupa).

3. (10 poena)

```
void arr_add (size_t arr_size) {
    static const size_t ELEM_SIZE = 4;
    static const size_t BLOCK_SIZE = (1<<14)/ELEM_SIZE; // 32KB/2/sizeof(int)
    static int a[BLOCK_SIZE], b[BLOCK_SIZE];

    size_t block = 0, remain = arr_size;

    while (remain>0) {
        size_t size = (remain>BLOCK_SIZE)?BLOCK_SIZE:remain;
        read_block(block*BLOCK_SIZE*ELEM_SIZE, size, &a);
        read_block(arr_size*ELEM_SIZE+block*BLOCK_SIZE*ELEM_SIZE, size, &b);

        for (size_t i=0; i<size; i++) a[i] += b[i];

        write_block(2*arr_size*ELEM_SIZE+block*BLOCK_SIZE*ELEM_SIZE, size, &a);

        remain = (remain>size)?remain-size:0;
        block++;
    }
}
```