Operativni Sistemi

NASM

Alati

- DOSBox
- NASM (Netwide Assembler)
- Notepad++

Okruženje (NASM)

- Skinuti nasm za DOS (`www.nasm.us -> download -> 2.12 -> dos -> nasm-2.12-dos.zip`)
- Napraviti direktorijum koji će da nam predstavlja `c:` pod DOS-om (npr `e:\os\dos`) i u njega raspakovati nasm (recimo u direktorijum `nasm`)

Okruženje (DOSBox)

- Skinuti DOSBox (`www.dosbox.com`). Instalirati na proizvoljnu lokaciju.
- Napomena: DOSBox je moguće raspakovati direktno kao da je arhiva, tako admin privilegija ne mora da bude neophodna.
- Pokrenuti konfiguracioni skript (`DOSBox 0.74 Options.bat`)
- Otvoriće se tekstualna datoteka. Na kraj dodati linije:

```
mount c e:\os\dos
path=z:\;c:\nasm
```

- Po potrebi zameniti `e:\os\dos` sa putanjom vašeg dos direktorijuma i `c:\nasm` sa putanjom unutar tog direktorijuma u kojoj je instaliran nasm

DOS komandna linija

- `cd` (change directory)
  - Menja trenutni aktivni direktorijum. Trenutni aktivni direktorijum je naveden levo od prompta
  - Ako se nalazimo na root `c:` particije, u kojoj se nalazi direktorijum `nasm`:
  - `C:\>cd nasm` (menja trenutni aktivni direktorijum na `nasm`)
  - Komanda `cd..` nas vraća na prethodni direktorijum
- Promena aktivne particije
  - Vrši se samo upisivanjem slova i dvotačke, npr
  - `c:`
  - Bitno: nije `cd c:`
- `dir` (directory)
  - Ispisuje sadržaj trenutno aktivnog direktorijuma

DOS komandna linija (nastavak)

- Taster “strelica na gore” nas vraća na prethodnu komandu. Ovako se možemo kretati kroz istoriju komandi
  - Korisno u slučaju da nas mrzi da kucamo neku prilično dugačku komandu koju smo već ranije iskucali – vidi naredni slajd :)
- Taster `TAB` je „autocomplete“ funkcija za imena fajlova i direktorijuma. Primera radi, ako kucamo:
- `cd <TAB>`
- Biće ponuđen prvi direktorijum u trenutnom (po abecednom redu). Svaki naredni pritisak na `TAB` će listati direktorijume.

Kompajlovanje

- Vrši se iz command prompta
  - DOSBox (DOS okruženje)
  - Start → Run → cmd (Windows okruženje)
- Podesiti aktivni direktorijum onaj u kojem se nalazi .asm fajl
- U komandi za kompajlovanje razmaci razdvajaju bitne elemente.
- Oblik komande je sledeći:
- `nasm test.asm -f bin -o test.com`
  - `nasm` : komanda
  - `test.asm` : naziv ulazne (izvorne) datoteke
  - `-f bin` : format izlaza
  - `-o test.com` : naziv izlazne (izvršne) datoteke

Primer 1

- `org 100h`
  - Navodimo adresu od koje treba da krene izvršavanje; vrednost `100h` je posebna – kod COM programa se izvršni kod uvek nalazi na toj adresi.
- `mov ah,9`
  - Premeštamo vrednost `9` u gornji deo registra `ax`.
- `mov dx,poruka`
  - Premeštamo u registar `dx` adresu poruke koju želimo da ispišemo.

Primer 1 (nastavak)

- `int 21h`
  - Pozivamo prekid (interrupt) koji ispisuje poruku (bitno: `ah` je `9`). Prekid `21h` sa argumentom ah=9h očekuje da se na adresi dx nalazi poruka. Ispisivanje se vrši redom, počevši od te adrese, sve dok se ne naiđe na karakter `"$"`.
- `ret`
  - Označava kraj programa
- `poruka: db 'Moj prvi program.$'`
  - Definišemo poruku kao niz sekvencijelnih bajtova i smeštamo je na adresu koju označavamo simbolom `"poruka"`

Registri

- Memorijske jedinice koje su direktno vidljve procesoru – da bismo obavili bilo koju operaciju, podatak sa kojim radimo mora da bude u registru.
- Registri opšte namene:
  - `ax, bx, cx, dx`
  - Dvobajtni su, ali se gornjem i donjem bajtu može pristupiti preko:
    - `ah, bh, ch, dh`
    - `al, bl, cl, dl`
- Par specijalnih registara:
  - `ip` – instruction pointer (sadrži adresu trenutne instrukcije)
  - `sp` – stack pointer (sadrži adresu vrha steka)

Podaci

- Pseudo-instrukcije:
  - `db` : define byte (podatak koji definišemo se zapisuje kao bajt)
  - `dw` : define word (… dvobajt)
  - `dd` : define double word (… četvorobajt)
  - `dq` : define quad word (… osmobajt)
- Zovemo ih “pseudo-” zato što se ne prevode na prave instrukcije procesora.
- Kompajler na poziciji ovih instrukcija smešta bukvalno podatke koje smo naveli.
- `db 'a',13,0` : u memoriji će se naći redom ASCII kod za 'a', broj 13 i broj 0.

Prekidi

- Primitivno event-driven programiranje.
- Osnovna ideja je da postoje događaji na koje želimo da reagujemo, i to tako što ćemo da prekinemo šta god da je trenutno aktuelno na procesoru, obradimo reakciju, i potom se vratimo na ono što je procesor bio radio pre prekidanja.
- Za prekide su definisani standardizovani kodovi koje svi moraju da poštuju (inače sistem nema smisla).
- Rutina koja obrađuje prekid se često naziva “handler”.

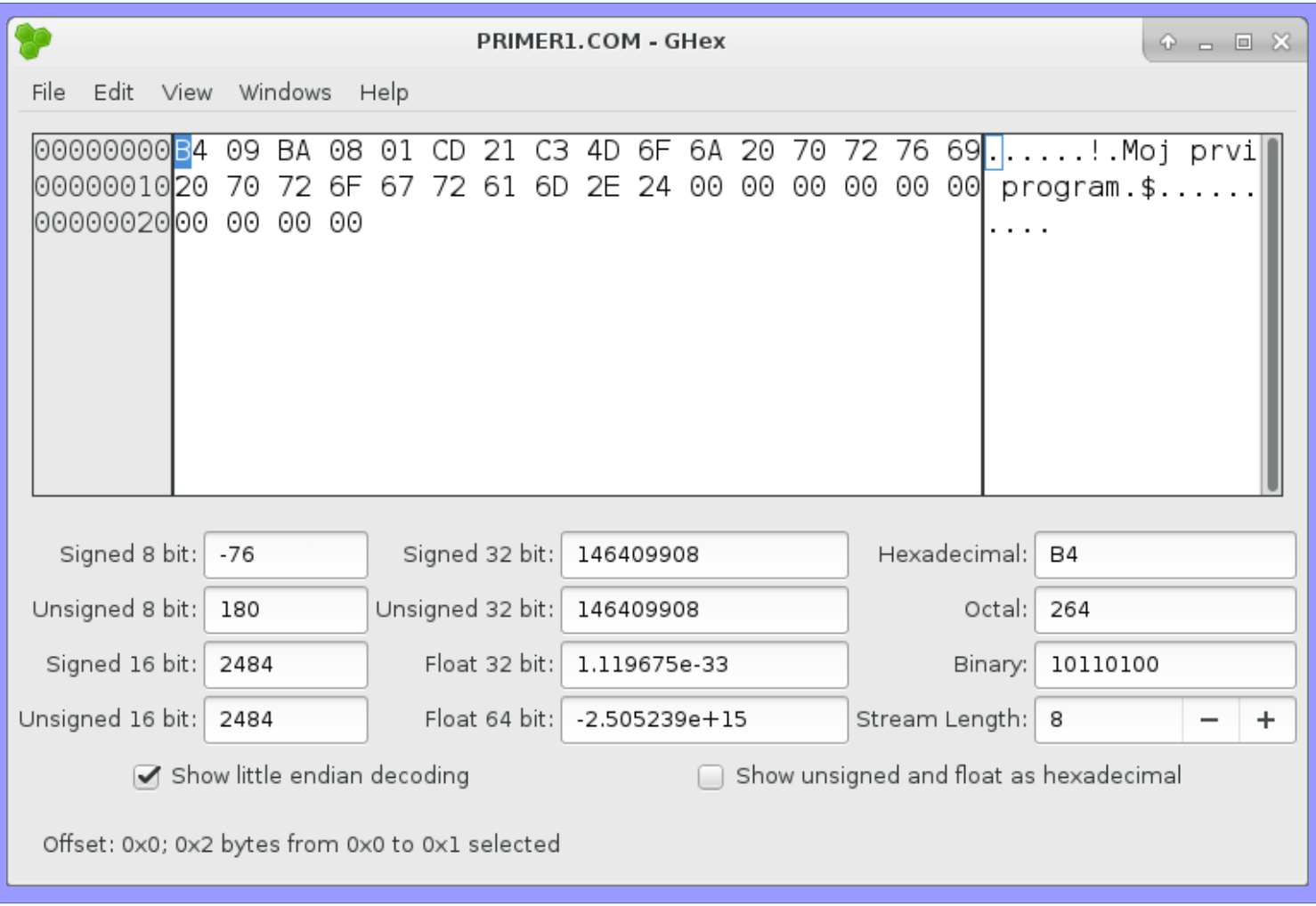
Prekidi

- Par primera prekida:
  - `10h` – BIOS prekidi ekrana
  - `16h` – BIOS prekidi tastature
  - `08h` – BIOS tajmerski prekid
  - `1Ch` – korisnički tajmerski prekid
  - `21h` – DOS sistemski prekid
- Prekidi `10h`, `16h` i `21h` obavljaju različite poslove. Izbor rutine se vrši na osnovu vrednosti koja se nalazi u registru `ah`. Drugi primeri za `ah`:
  - `ah = 2Ch` – čitanje sistemskog vremena
  - `ah = 3Ch` – kreiranje datoteke
  - itd.

COM program

- Kompajlovan program u najjednostavnijoj formi ima instrukcije i podatke, zapisane sekvencijalno. COM programi ovako izgledaju na disku, kao i u memoriji.
- Spisak opcode-ova.

Prethodni primer, kompajlovan



org 100h dopuna

- MS-DOS pri učitavanju COM programa u memoriju uvek stavlja izvršni kod i podatke (sve sa prethodnog slajda) na adresu `100h`. Pre te adrese se nalazi tzv Program Segment Prefix, koji je svojevrsno zaglavlje programa, i trenutno nam nije značajno.
- Ali ovo bi značilo da instrukcija "`mov dx, poruka`" imala pogrešan argument, ako ne bismo uzeli ovaj pomeraj u obzir.
- Instrukcija `org` translira sve ovakve relativne adrese za navedeni argument. Mi ovde igrom slučaja znamo da je to tačno `100h`, zato što je taj pomeraj uvek takav kod COM programa. Dakle, naša adresa `0008h` postaje `0108h`.

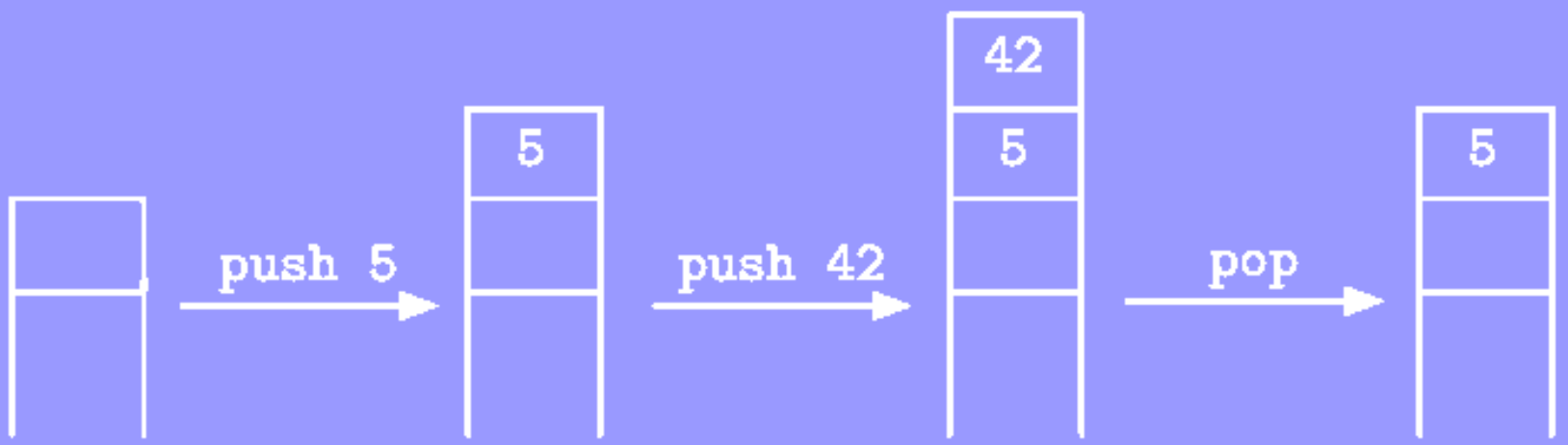
Primer 2

- `int 16h`
  - Prekid za rad sa tastaturom. Kad `ah` ima vrednost `0`, uneseni ASCII karakter se upisuje u `al`. Ovaj poziv je blokirajući (program čeka da se pritisne taster na tastaturi).
- `int 10h`
  - Prekid za rad sa ekranom. Kad `ah` ima vrednost `0eh`, na trenutnu poziciju kursora se ispisuje ASCII karakter koji se nalazi u `al`.

Primer 2 (nastavak)

- `cmp al, ESC`
  - Poredi vrednost u `al` sa konstantom `1bh` (definisana na vrhu programa). Rezultat poređenja se beleži kao skup flag-ova koji mogu da se naknadno testiraju.
- `je izlaz`
  - „Jump if equal“ - skače na labelu “`izlaz`” ako je postavljen “`e`” flag, koji predstavlja jednakost argumenata u prethodnom poređenju.
- `jmp citaj`
  - Bezuslovni skok na labelu “`citaj`”, čime se efektivno pravi petlja.

Stek



Stek

- Svaki proces („pokrenut program“) ima svoj stek.
- Stek je struktura podataka kod koje su dozvoljene dve operacije:
  - `push` – stavljanje podatka na vrh steka
  - `pop` – skidanje podatka sa vrha steka
- Kod DOS-a, stek je uvek dvobajtni, tj. na stek se smeštaju i sa steka se skidaju uvek po dva bajta.

Stack pointer

- Stack pointer (`sp`) je registar posebne namene koji sadrži trenutni pokazivač na stek. `sp` pokazuje na poslednju push-ovanu vrednost, ne ispred nje.
- Termin “pokazuje” može da bude malo nejasan. Formalno, “pokazivanje na vrednost `x`” se odnosi na ideju da registar `sp` sadrži adresu vrednosti `x`.
- Stek raste “na dole” u memoriji, tako da
  - Inkrementiranjem `sp` „skidamo“ vrednosti sa steka.
  - Dekrementiranjem `sp` „stavljamo“ vrednosti na stek.

Pozivi (primer 3)

- Instrukcija `call` se koristi za „poziv funkcije“. Zapravo, ono što se dešava je bezuslovni skok na labelu, s time da se, pre skoka, na stek stavlja adresa sa koje je izvršen poziv.
- Funkcija se završava izvršavanjem instrukcije `ret`.
  - Instrukcija `ret` automatski sa steka skida dva bajta i vrši bezuslovni skok na tu adresu.
- Napomena: niko ne vrši proveru koja dva bajta su u pitanju. Stek mora da ostane „čist“, tj. sve što se stavi na stek mora i da se skine sa njega. U suprotnom će doći do problema pri izvršavanju instrukcije `ret`.

Zadaci za vežbu

- Izmeniti drugi primer tako da se unesena velika slova ispisuju kao mala (nije bitno šta se desi sa malim slovima)
- Izmeniti drugi primer tako da se unesena mala slova ispisuju kao velika (nije bitno šta se desi sa velikim slovima)
- Izmeniti drugi primer tako da se unesena mala slova ispisuju kao velika, i velika kao mala