# Uvod v računalništvo (UvR)

# Sistemska programska oprema in navidezni stroji

Danijel Skočaj
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Literatura: Invitation to Computer Science, poglavje 6

v1.0
Št. leto 2013/14

# Cilji predavanja

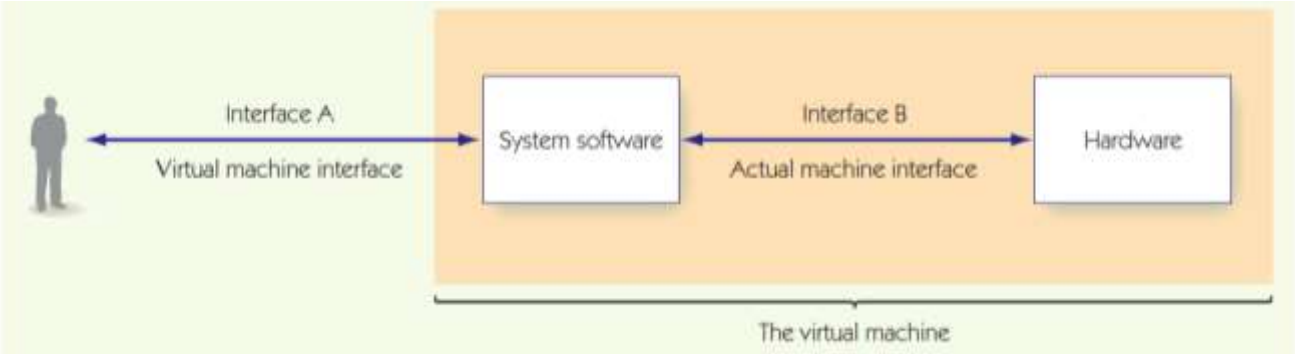
- Opisati glavne kategorije sistemskih nalog
- Primerjati navidezni stroj z golim strojem
- Razložiti prednosti pisanja sistemske programske opreme v zbirniku namesto v strojnem jeziku
- Brati in pisati kratke programe v zbirniku
- Opisati kako zbirnik prevaja programe v zbirnem jeziku v strojne ukaze
- Navesti in opisati pet temeljnih nalog operacijskega sistema
- Opisati različne generacije operacijskih sistemov

# Uvod

- Gol stroj
  - na nivoju strojne opreme
  - pisanje binarnih ukazov
  - pisanje podatkov z dvojiškimi pštevili/kodami
  - nalaganje ukazov v pomnilnik
  - zagon programa s programskim števcem
- Zelo prijazno za stroj
- Zelo neprijazno (nemogoče) za človeka
- Potrebno je zgraditi vmesnik med strojem in računalnikom
  - ki bo skril nepotrebne strojne podrobnosti
  - bo omogočil človeku bolj razumljivo uporabo računalnika

# Sistemska programska oprema

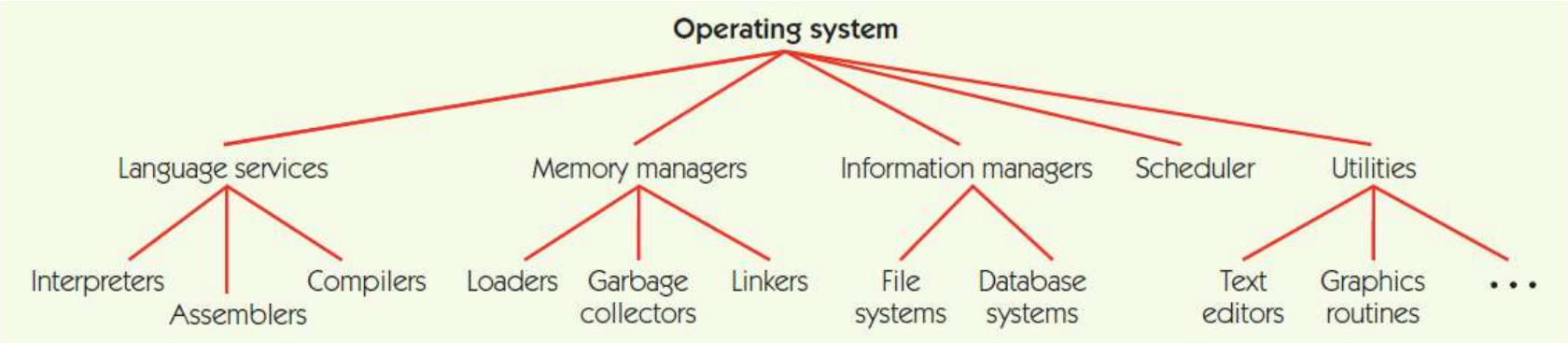
- Vmesnik med človekom in strojno opremo




- Navidezni stroj: množica storitev in virov, kot jih človeku predstavi sistemska programska oprema
- Naloge sistemske programske opreme:
  - skriti kompleksne in nepomembne detajle notranje strukture Von Neumannove arhitekture
  - predstaviti uporabniku pomembne informacije na razumljiv način
  - dovoliti uporabniku dostop do virov na enostaven in učinkovit način
  - zagotoviti varno okolje za delovanje računalnika

# Sistemska programska oprema

- Operacijski sistem
  - komunikacija z uporabnikom
  - zaganja ostale sistemske programe in aplikacije
- (Grafični) uporabniški vmesnik
- V/I sistemi: komunikacija z raznimi napravami
- Jezikovni servisi: podpora za višje-nivojske programske jezike
- Upravljavci s pomnilnikom: alokacija pomnilnika za programe
- Upravljavci z informacijami: upravljanje z množico podatkov
- Razporejevalnik: upravlja s programi, ki naj bi se izvedli
- Pripomočki: orodja, programske knjižnice



# Navidezni stroj

Gol stroj:

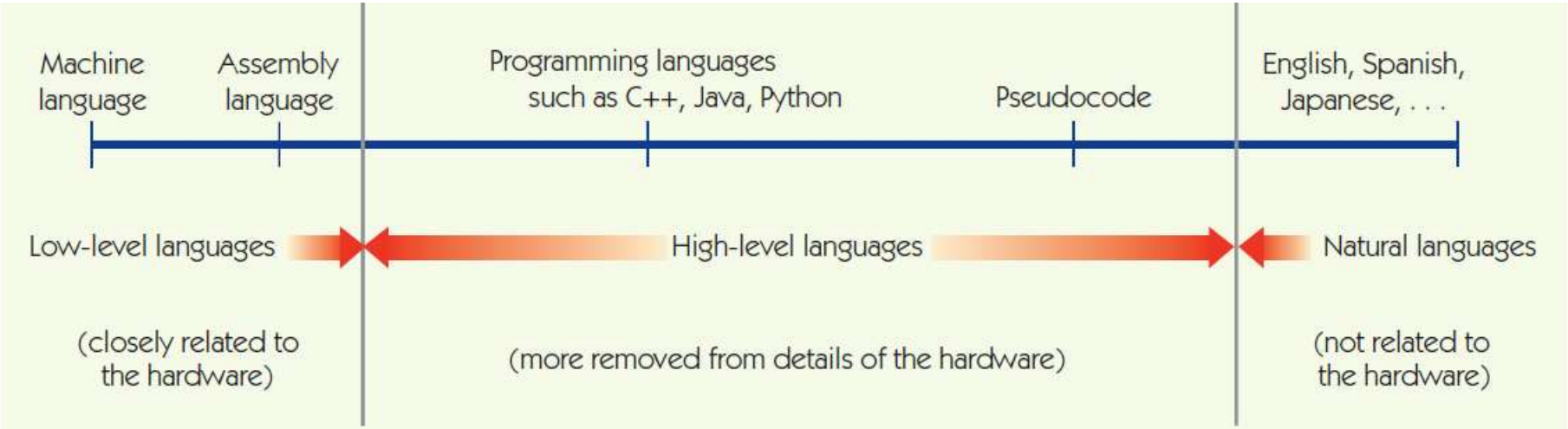
1. Napiši program v dvojiškem sistemu
2. Naloži ukaze enega za drugim v pomnilnik
3. Vstavi začetni ukaz na pomnilniško lokacijo 0 in zaženi delovanje
4. Preberi rezultate enega za drugim v dvojiškem sistemu

Navidezni stroj:

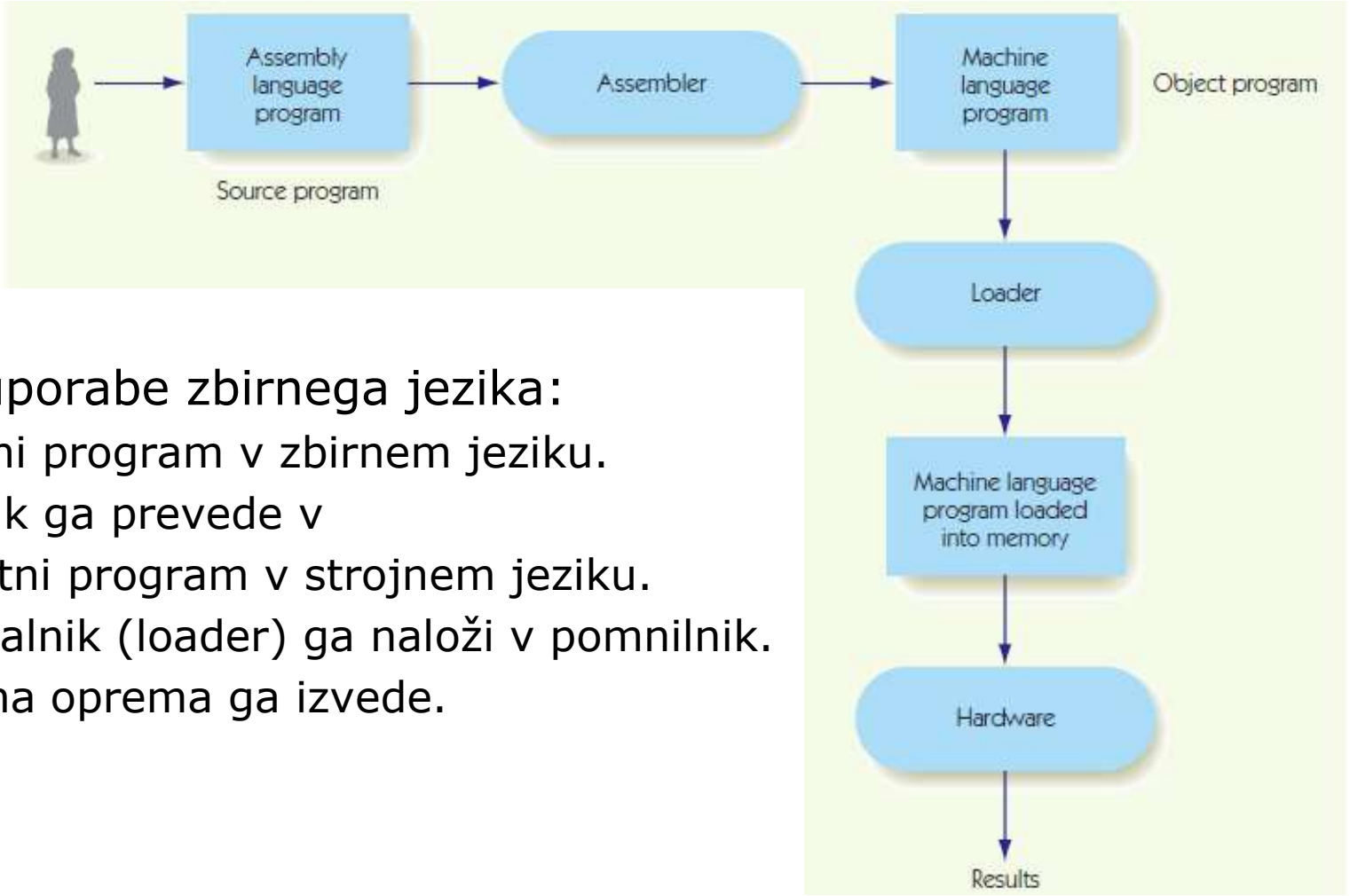
1. Napiši program v visoko-nivojskem programskem jeziku z urejevalnikom besedil
2. Shrani program v mapo
3. Prevedi program v dvojiški strojni jezik
4. Uporabi razporejevalnik za nalaganje in zagon programa
5. Uporabi V/I sistem za izpis rezultatov

# Programski jeziki

- Zbirni jezik (asembler)
  - Ukazi se preslikajo neposredno v strojne ukaze (eden v enega)
  - Simbolične kode ukazov (ne binarne)
  - Simbolični naslovi za ukaze in podatke
  - Pseudo ukazi z uporabniku prijaznejšimi servisi (npr. generiranje podatkov)
- Nizko-nivojski programski jeziki: strojni jezik, zbirni jezik
- Visoko-nivojski programski jeziki: Java, C, Python



# Zbirnik




- Proces uporabe zbirnega jezika:
  - Izvorni program v zbirnem jeziku.
  - Zbirnik ga prevede v
  - objektni program v strojnem jeziku.
  - Nalagalnik (loader) ga naloži v pomnilnik.
  - Strojna oprema ga izvede.

# Format zbirnega jezika

```
NEXTSTEP:    LOAD    X        --Put X into reg. R
```

- Oznaka: opcijsko simbolično ime za pomnilniško lokacijo
- Simbolično ime (okrajšava, mnemonik) za strojni ukaz
- Naslov pomnilniške lokacije
- -- komentar

- Veliko bolj razumljivo kot binarni ukazi!

# Nabor ukazov: primer

| Binary Op Code | Operation | Meaning |
|---|---|---|
| 0000 | LOAD X | CON(X) → R |
| 0001 | STORE X | R → CON(X) |
| 0010 | CLEAR X | 0 → CON(X) |
| 0011 | ADD X | R + CON(X) → R |
| 0100 | INCREMENT X | CON(X) + 1 → CON(X) |
| 0101 | SUBTRACT X | R – CON(X) → R |
| 0110 | DECREMENT X | CON(X) – 1 → CON(X) |
| 0111 | COMPARE X | if CON(X) > R then GT = 1 else 0<br>if CON(X) = R then EQ = 1 else 0<br>if CON(X) < R then LT = 1 else 0 |
| 1000 | JUMP X | Get the next instruction from memory location X. |
| 1001 | JUMPGT X | Get the next instruction from memory location X if GT = 1. |
| 1010 | JUMPEQ X | Get the next instruction from memory location X if EQ = 1. |
| 1011 | JUMPLT X | Get the next instruction from memory location X if LT = 1. |
| 1100 | JUMPNEQ X | Get the next instruction from memory location X if EQ = 0. |
| 1101 | IN X | Input an integer value from the standard input device and store into memory cell X. |
| 1110 | OUT X | Output, in decimal notation, the value stored in memory cell X. |
| 1111 | HALT | Stop program execution. |

# Prednosti zbirnega jezika

- Prednosti pred strojnim jezikom:
  - jasnost
  - berljivost
  - vzdrževalnost
  - pseudo ukazi:
    - ukazi, ki niso neposredno prevedeni v strojne ukaze:
    - .BEGIN in .END označujeta začetek in konec programa za prevajanje – za generiranje programa
    - .DATA pretvori podatke v desetiškem sistemu v dvojiški sistem in jih shrani na navedeni lokaciji – generiranje podatkov
- Tipična struktura programa v zbirnem jeziku:

```
.BEGIN      --This must be the first line of the program
    :       --Assembly language instructions like those in Figure 6.5
  HALT      --This instruction terminates execution of the program
    :       --Data generation pseudo-ops such as
            --.DATA are placed here, after the HALT
.END        --This must be the last line of the program
```

# Primeri

- Primer:

```
x=y+3
```

```
LOAD Y
ADD THREE
STORE X
…          -- Data comes after .END
X:         .DATA 0  -- X is initially 0
Y:         .DATA 5 – Y is initially 5
THREE: .DATA 3 –The constant 3
```

- Primer:

```
input a and b
while a > b do
        print a
        a = a – 2
```

```
IN A
IN B
LOOP1: LOAD A                    A:  .DATA 0
           COMPARE B             B:  .DATA 0
           JUMPLT LOOP1END       TWO: .DATA 2
           JUMPEQ LOOP1END
           OUT A
           LOAD A
           SUBTRACT TWO
           STORE A
           JUMP LOOP1
LOOP1END: …
```

# Primer programa

- Vsota števil

| Step | Operation |
|---|---|
| 1 | Set the value of *Sum* to 0 |
| 2 | Input the first number *N* |
| 3 | While *N* is not negative do |
| 4 | Add the value of *N* to *Sum* |
| 5 | Input the next data value *N* |
| 6 | End of the loop |
| 7 | Print out *Sum* |
| 8 | Stop |

```
            .BEGIN                  --This marks the start of the program
            CLEAR     SUM           --Set the running sum to 0 (line 1)
            IN        N             --Input the first number N (line 2)
--The next three instructions test whether N is a negative number (line 3)
AGAIN:      LOAD      ZERO          --Put 0 into register R
            COMPARE   N             --Compare N and 0
            JUMPLT    NEG           --Go to NEG if N < 0
--We get here if N ≥ 0. We add N to the running sum (line 4)
            LOAD      SUM           --Put SUM into R
            ADD       N             --Add N. R now holds (N + SUM)
            STORE     SUM           --Put the result back into SUM
--Get the next input value (line 5)
            IN        N
--Now go back and repeat the loop (line 6)
            JUMP      AGAIN
--We get to this section of the program only when we encounter a negative value
NEG:        OUT       SUM           --Print the sum (line 7)
            HALT                    --and stop (line 8)
--Here are the data generation pseudo-ops
SUM:        .DATA     0             --The running sum goes here
N:          .DATA     0             --The input data are placed here
ZERO:       .DATA     0             --The constant 0
--Now we mark the end of the entire program
            .END
```

# Prevajanje in nalaganje

- Zbirnik prevede program v zbirnem jeziku v strojni jezik
  - pretvori simbolične kode ukazov v strojne ekvivalente
  - pretvori simbolične oznake v naslove pomnilniških lokacij
  - izvede pseudo ukaze
  - zapiše objektno (izvršljivo) datoteko s strojnimi ukazi

- Nalagalnik pripravi program na izvajanje
  - naloži ukaze v pomnilnik
  - sproži strojno opremo, da začne izvajati program

# Tabela kod ukazov

- Pretvorba simboličnih kod ukazov
  - tabela simboličnih kod ukazov ter binarnih ekvivalentov
  - zbirnik poišče ustrezne kode ukazov in jih zamenja z binarnimi
  - uporablja binarno iskanje za pospešitev iskanja

| Operation | Binary Value |
|---|---|
| ADD | 0011 |
| CLEAR | 0010 |
| COMPARE | 0111 |
| DECREMENT | 0110 |
| HALT | 1111 |
| ⋮ | |
| STORE | 0001 |
| SUBTRACT | 0101 |

# Tabela simbolov

- Pretvorba simbolnih oznak v naslove pomnilniških lokacij
- Zbirnik potrebuje dva koraka
  1. Zgradi tabelo simbolov

| Label | Code | | Location Counter |
|---|---|---|---|
| LOOP: | IN | X | 0 |
| | IN | Y | 1 |
| | LOAD | X | 2 |
| | COMPARE | Y | 3 |
| | JUMPGT | DONE | 4 |
| | OUT | X | 5 |
| | JUMP | LOOP | 6 |
| DONE: | OUT | Y | 7 |
| | HALT | | 8 |
| X: | .DATA | 0 | 9 |
| Y: | .DATA | 0 | 10 |

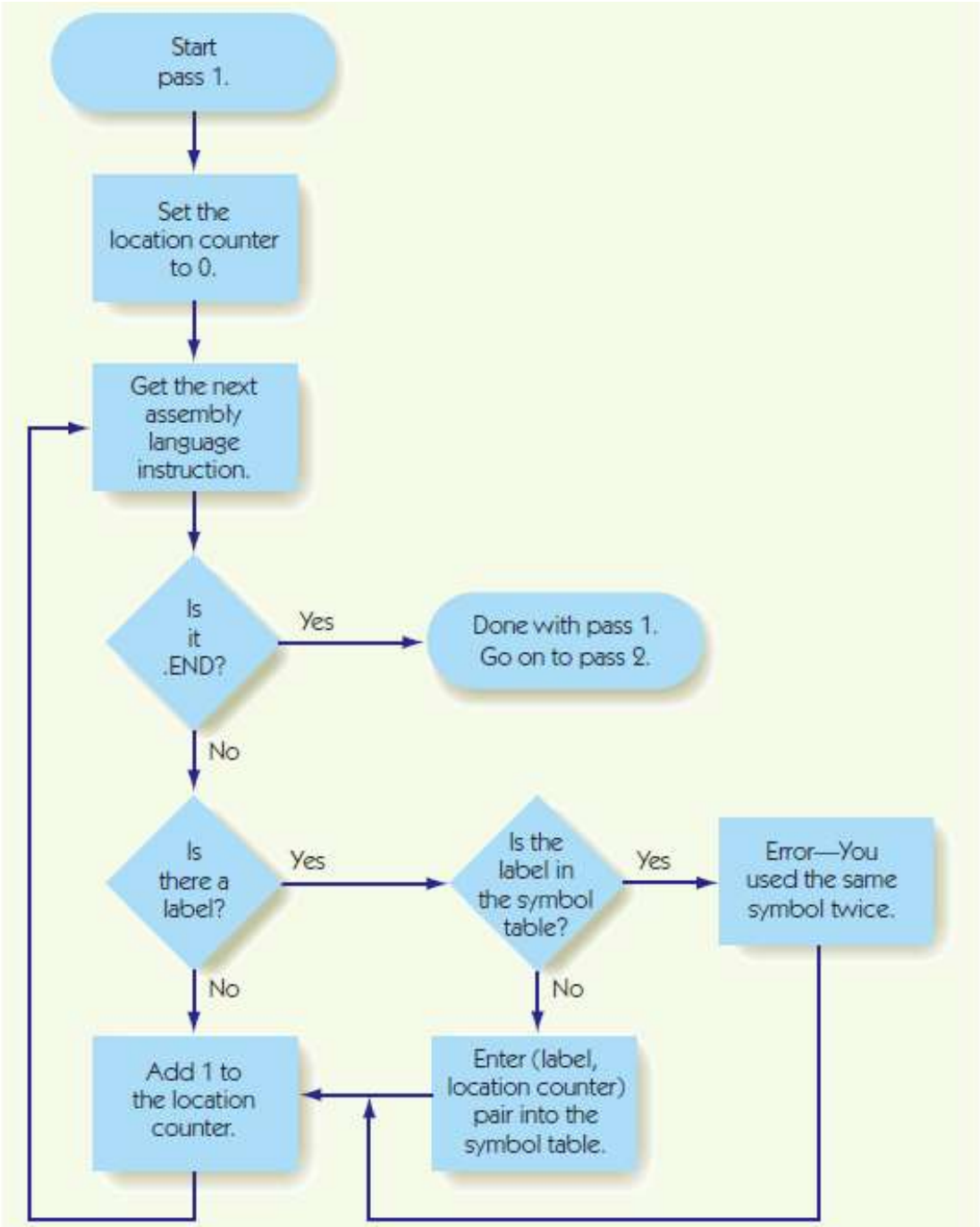
Symbol Table

| Symbol | Address Value |
|---|---|
| LOOP | 0 |
| DONE | 7 |
| X | 9 |
| Y | 10 |

  2. Simbole nadomesti z binarnimi vrednostmi iz tabele simbolov

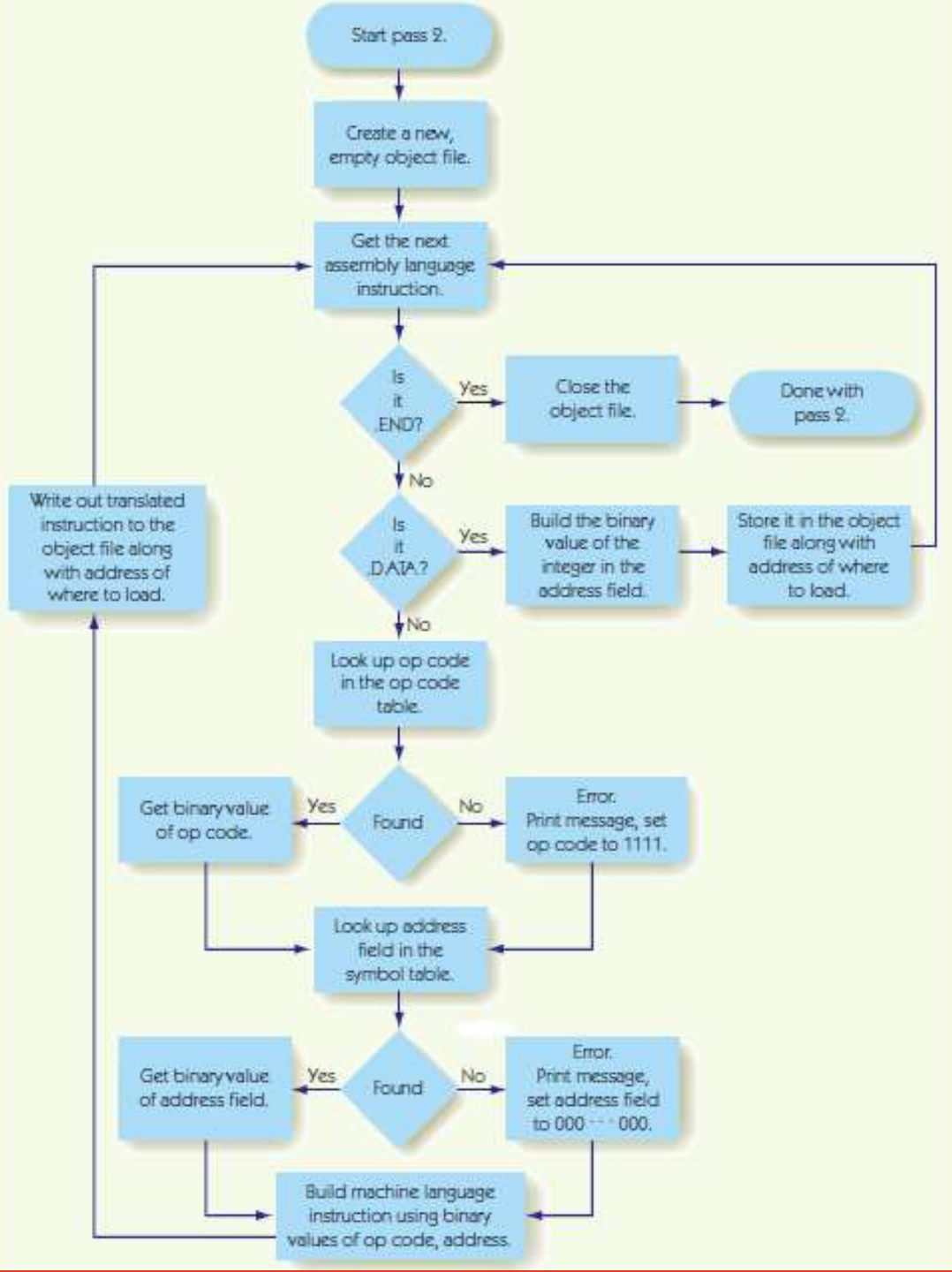
# Prvi korak

- Gradnja tabele simbolov



# Drugi korak

- Poišče in zamenja simbolične kode ukazov z binarnimi
- Zamenja simbolne oznake z naslovi pomnilniških lokacij iz tabele simbolov
- Izvede pseudo ukaze .DATA in na ustrezne naslove zapiše ustrezne binarne vrednosti
- Zapiše ukaze v objektno (izvršljivo) datoteko



# Primer

- Primer programa v strojnem jeziku

| **Instruction Format:** | **Op Code** | **Address** |
|---|---|---|
| | 4 bits | 12 bits |

**Object Program:**

| *Address* | *Machine Language Instruction* |
|---|---|
| 0000 | 1101 000000001001 |
| 0001 | 1101 000000001010 |
| 0010 | 0000 000000001001 |
| 0011 | 0111 000000001010 |
| 0100 | 1001 000000000111 |
| 0101 | 1110 000000001001 |
| 0110 | 1000 000000000000 |
| 0111 | 1110 000000001010 |
| 1000 | 1111 000000000000 |
| 1001 | 0000 000000000000 |
| 1010 | 0000 000000000000 |

# Operacijski sistemi

- Tipični ukazi operacijskega sistema:
  - prevedi program
  - poženi program
  - shrani informacijo v datoteko
  - poišči predhodno shranjeno datoteko
  - izpiši podatke o datotekah
  - izbriši ali preimenuj datoteko
  - kopiraj datoteko iz ene V/I naprave na drugo
  - dovoli uporabniku spremeniti geslo
  - vzpostavi mrežno povezavo
  - sporoči trenuten čas in datum

# Glavne naloge operacijskih sistemov

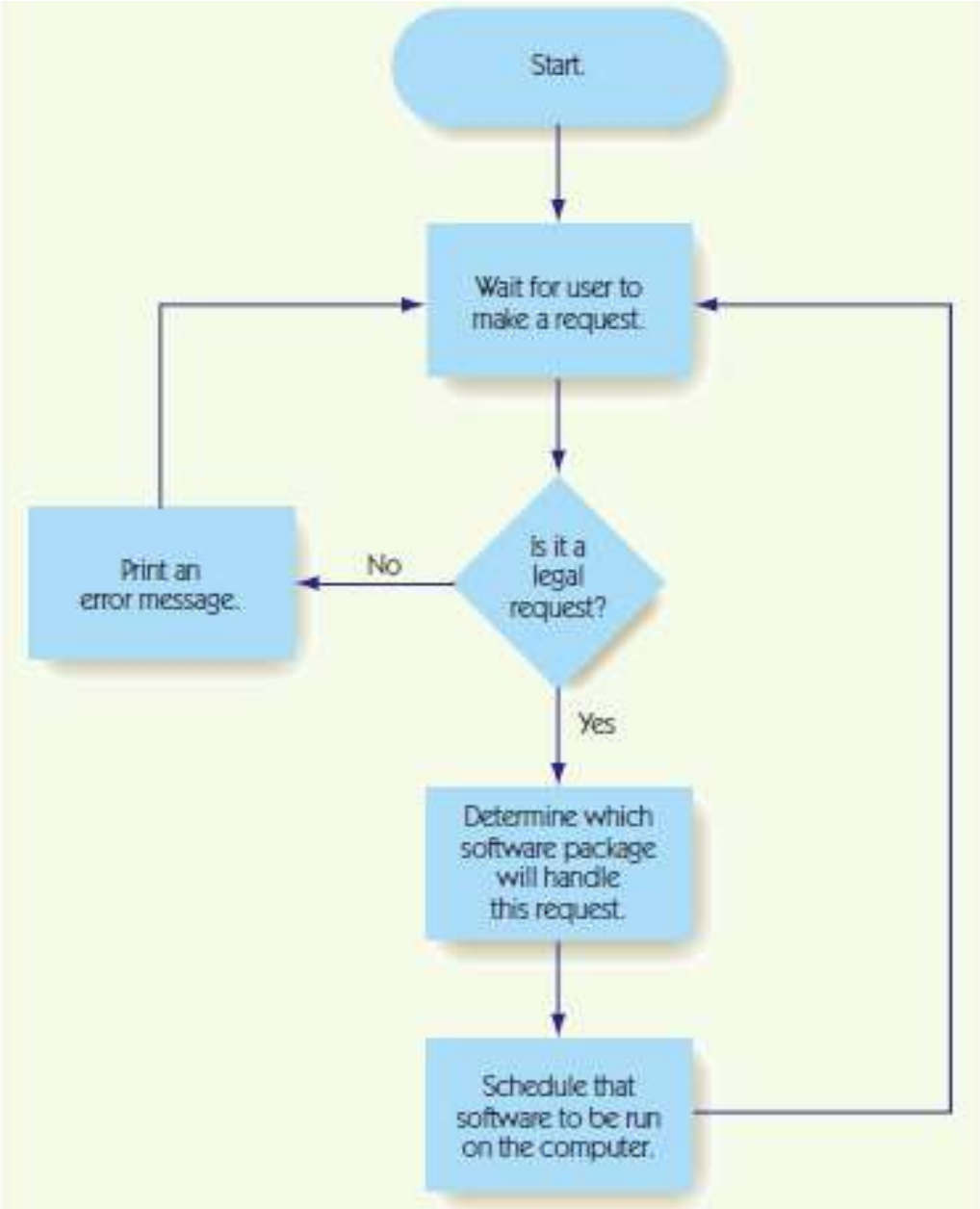
- Uporabniški vmesnik
- Dostop do sistema in datotek
- Varnost in zaščita sistema
- Učinkovito razporejanje virov
- Varna uporaba virov

- Procesiranje V/I
- Prioritetizacija programov
- Premikanje programov v in iz pomnilnika
- Upravljanje z napakami
- …

# Uporabniški vmesnik

- Uporabnik preko uporabniškega vmesnika komunicira z operacijskim sistemom (in računalnikom)
- Operacijski sistem sprejema ukaze in jih razporeja naprej

- Tekstovni
  - vnašanje ukazov preko tipkovnice
  - ukazni jezik
- Grafični
  - vnašanje ukazov na vizualen način



# Uporabniški vmesnik

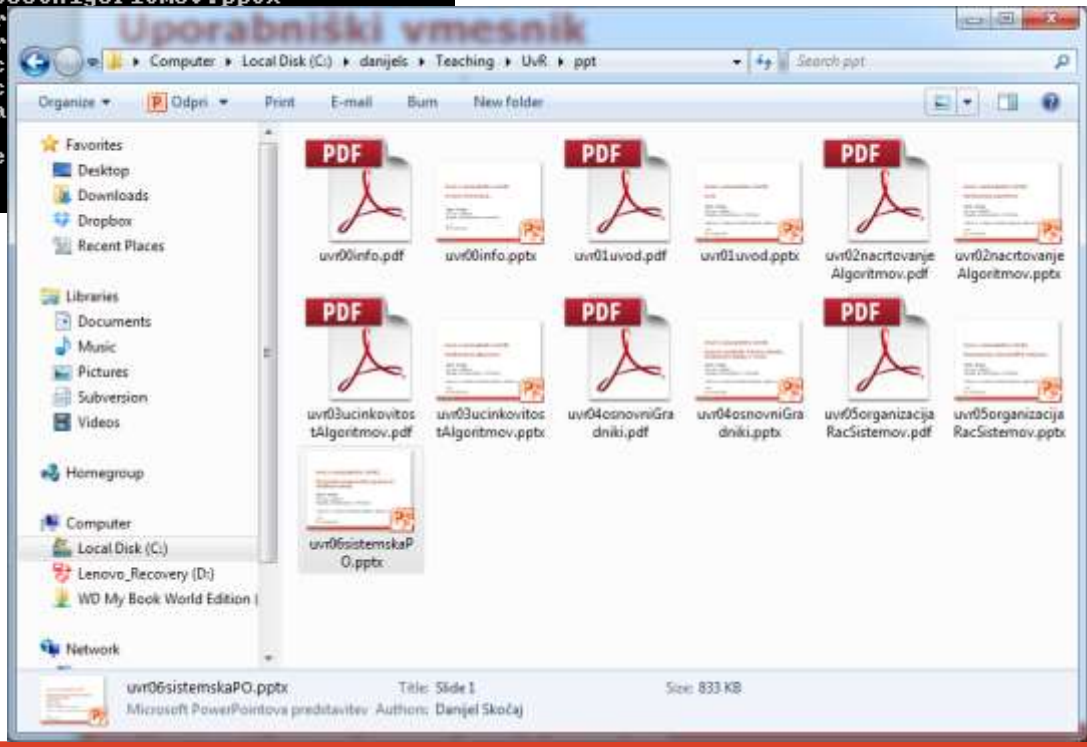
```
C:\danijels\Teaching\UvR\ppt>cd C:/danijels/teaching/UvR/ppt

C:\danijels\Teaching\UvR\ppt>dir
 Volume in drive C has no label.
 Volume Serial Number is 0412-993B

 Directory of C:\danijels\Teaching\UvR\ppt

05.11.2013  23:53    <DIR>          .
05.11.2013  23:53    <DIR>          ..
09.10.2013  07:44           257.261 uvr00info.pdf
08.10.2013  14:13           711.606 uvr00info.pptx
09.10.2013  07:45         1.291.563 uvr01uvod.pdf
09.10.2013  07:45         4.840.729 uvr01uvod.pptx
09.10.2013  07:46           928.201 uvr02nacrtovanjeAlgoritmov.pdf
09.10.2013  07:46         1.219.974 uvr02nacrtovanjeAlgoritmov.pptx
16.10.2013  12:53         1.034.257 uvr03ucinkovitostAlgoritmov.pdf
16.10.2013  12:53         1.795.000 uvr03ucinkovitostAlgoritmov.pptx
22.10.2013  23:07         1.529.150 uvr04osnovniGr
22.10.2013  23:06         3.967.087 uvr04osnovniGr
30.10.2013  01:21           903.432 uvr05organizac
30.10.2013  01:14         1.735.180 uvr05organizac
05.11.2013  23:53           853.822 uvr06sistemska
              13 File(s)     21.067.262 bytes
               2 Dir(s)  48.280.084.480 bytes free

C:\danijels\Teaching\UvR\ppt>
```



# Varnost in zaščita sistema

- Operacijski sistem dovoli samo avtorizirane posege oz. dostope do virov
- Dostop običajno dovoljen samo z uporabniškim imenom in geslom
  - individualna gesla
  - superuporabniki (superusers) imajo več pooblastil
  - podatki o geslih so običajno šifrirani
- Datoteke in direktoriji imajo sezname dovoljenih posegov:
  - beri datoteko (R)
  - dodaj novo informacijo v datoteko (A)
  - spremeni trenutno informacijo (C)
  - izbriši datoteko (D)

File: GRADES

| Name | Permitted Operations |
|---|---|
| Smith | R (R = Read only) |
| Jones | RA (A = Append) |
| Adams | RAC (C = Change) |
| Doe | RACD (D = Delete) |

# Učinkovito razporejanje virov

- Operacijski sistem učinkovito razporeja vire programov, da so čim bolje izkoriščeni
  - CPE naj bo čim bolj uporabljena
  - programi naj se čim bolj učinkovito izvajajo
- Več programov je lahko hkrati aktivnih
- Vsak program je v enem od treh stanj:
  - Se izvaja: program trenutno uporablja CPE
  - Čaka: program čaka, da se zaključijo V/I operacije
  - V pripravljenosti: program je pripravljen, da se izvede v CPE
- Operacijski sistem vzdržuje vrsto programov v pripravljenosti in ustrezno porazdeli vire

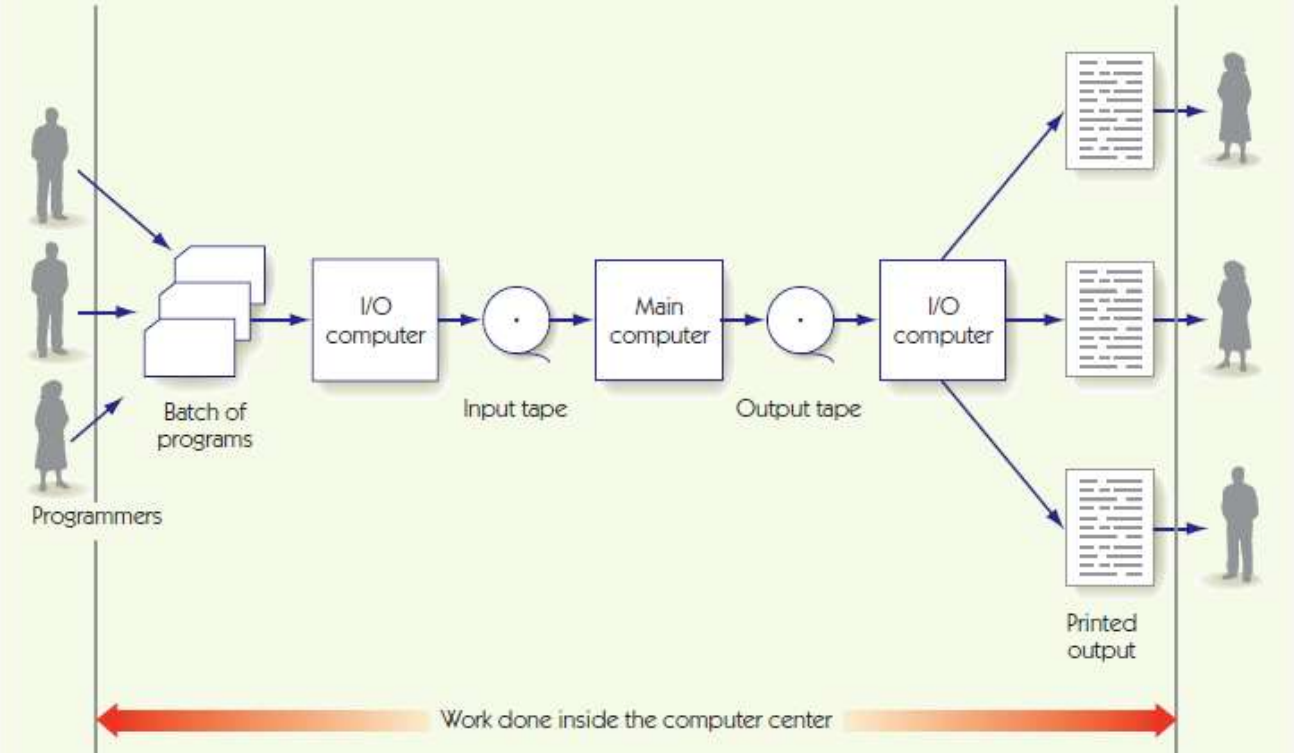
# Varna uporaba virov

- Zagotavljanje, da se računalnik ne ujame v mrtvo zanko
- Hkratno izvajanje več programov in dostopanje do virov
- Mrtva zanka nastopi, ko več programov zaseda nekaj virov, hkrati pa čaka na vire, ki jih zasedajo drugi programi

- Izogibanje mrtvim zankam
  - če ne moreš dobiti vseh zahtevanih virov, osvobodi vse vire, ki jih zasedaš in poizkusi ponovno pozneje
- Reševanje mrtvih zank
  - če se ne zgodi pričakovani dogodek, povprašaj ponovno

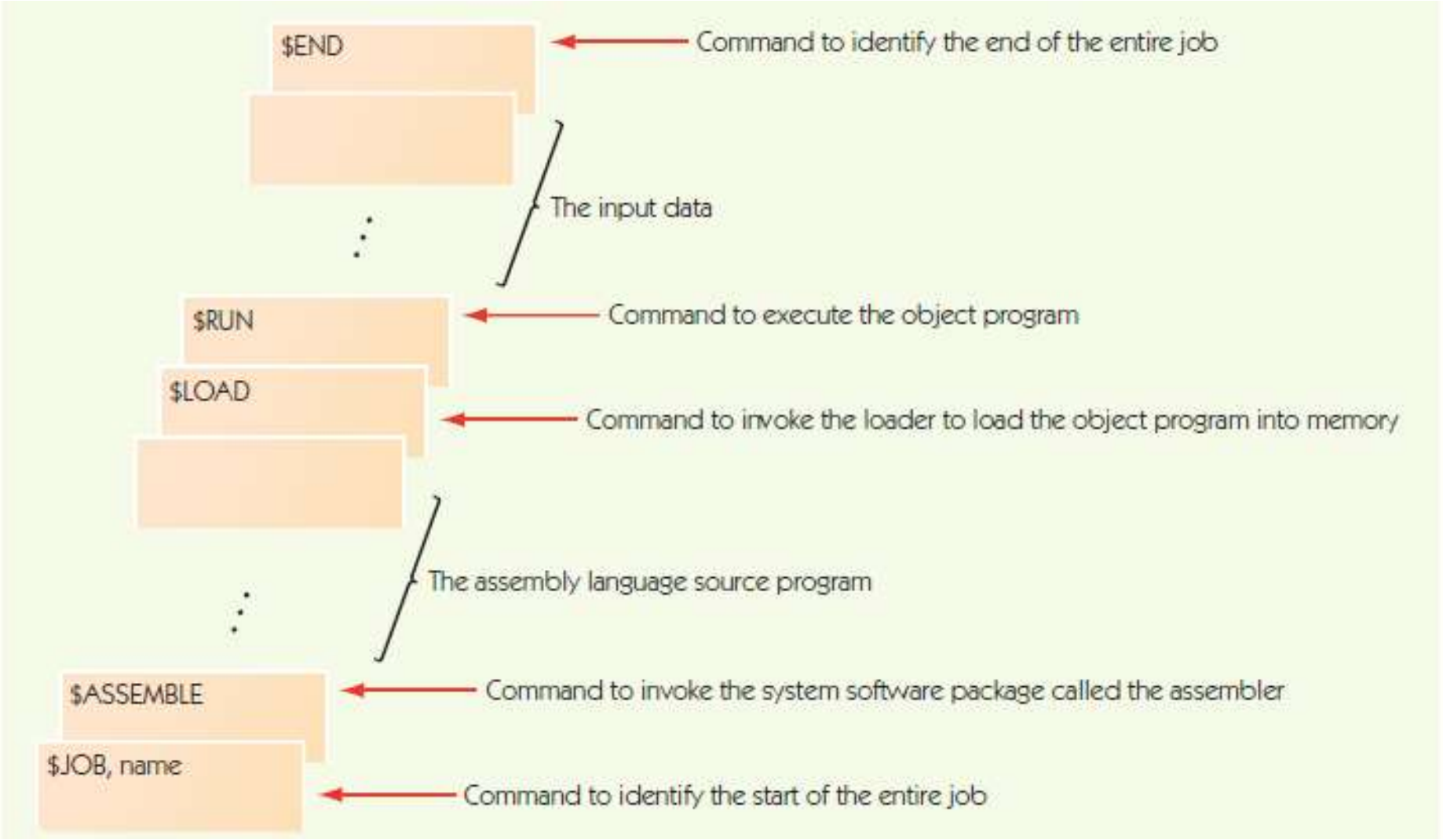
# Zgodovinski pregled – 1., 2. generacija

- Prva generacija (1945-1955): skoraj gol stroj
  - zelo skromni operacijski sistemi (ali celo brez)
  - programerji delali neposredno z računalniki
  - veliko neporabljenih virov
- Druga generacija (1955-1965): paketni operacijski sistemi
  - z računalnikom upravlja operater
  - boljša izkoriščenost virov
  - jezik za nadzor opravil



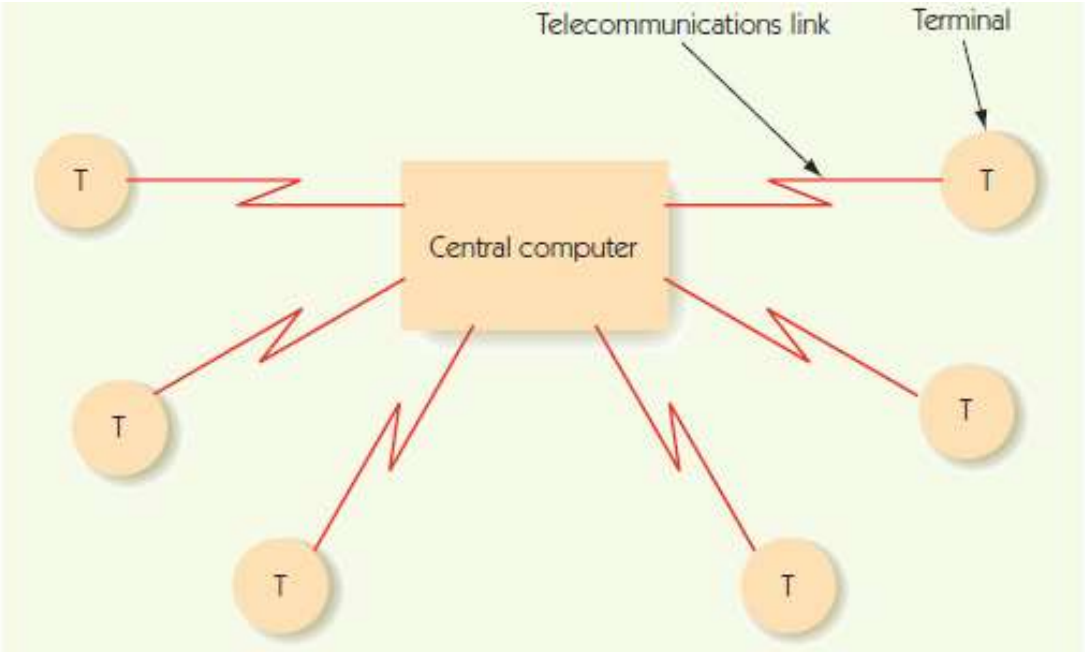
# Zgodovinski pregled – 2. generacija

- Struktura tipičnega paketa opravil



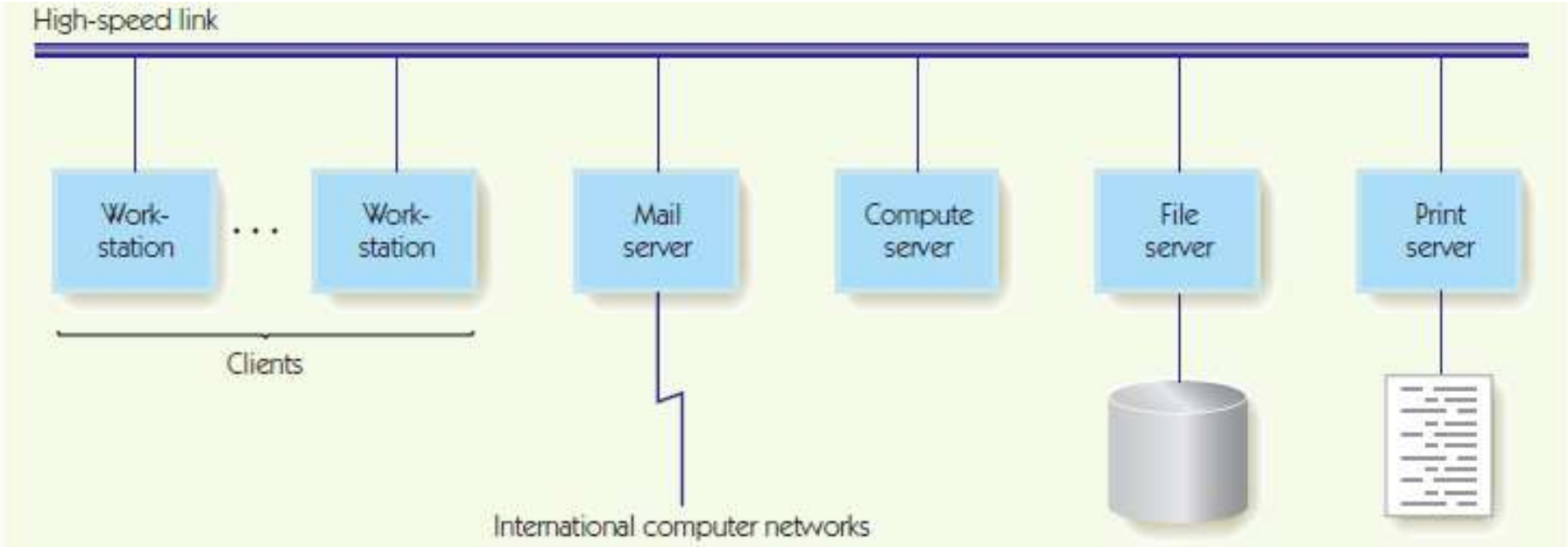
# Zgodovinski pregled – 3. eneracija

- Tretja generacija (1965-1985): večopravilni operacijski sistemi
  - Več programov hkrati naloženih v pomnilniku
  - Preklapljanje med programi med čakanjem na V/I naprave
  - Potreba po varnosti računalniških sistemov
    - priviligirane operacije dovoljene samo administratorju
    - omejena področja pomnilnika za posamezne programe
  - Delitev procesorskega časa med več uporabniki
    - princip centralnega računalnika in terminalov
    - interaktivno delo
    - časovne rezine
    - odzivni čas
  - Prvi OS za osebne računalnike



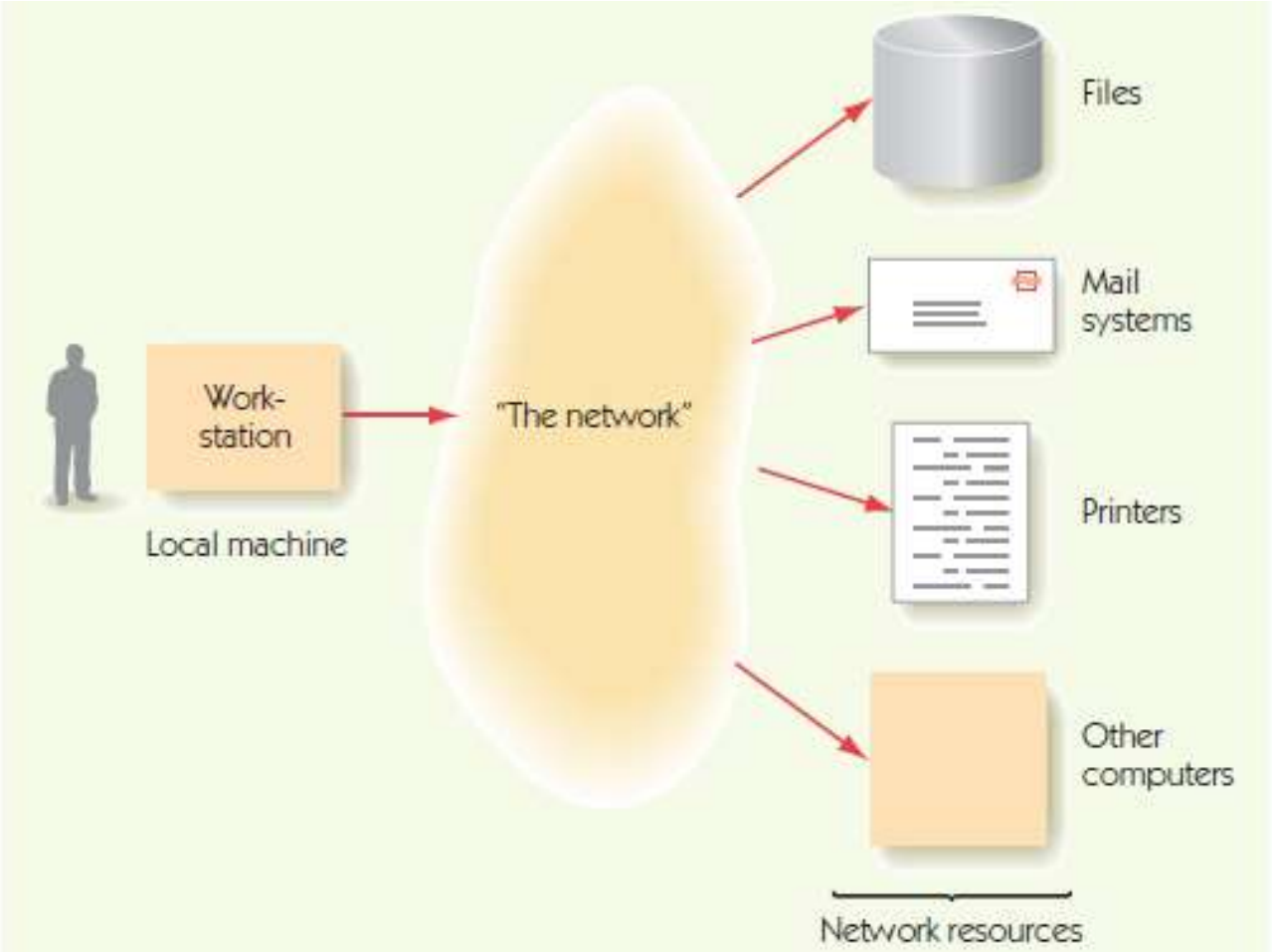
# Zgodovinski pregled – 4. generacija

- Četrta generacija (1985 - ): omrežni operacijski sistemi
  - operacijski sistem podpira vse lokalne storitve
  - podpira tudi storitve v omrežju:
    - skupni tiskalnik
    - strežniki: el. pošta, podatki, datoteke, splet
    - povezava na Internet
  - arhitektura odjemalec-strežnik
  - grafični uporabniški vmesniki



# Zgodovinski pregled – 4. generacija

- Razširjeni navidezni stroj



# Zgodovinski pregled – 4. generacija

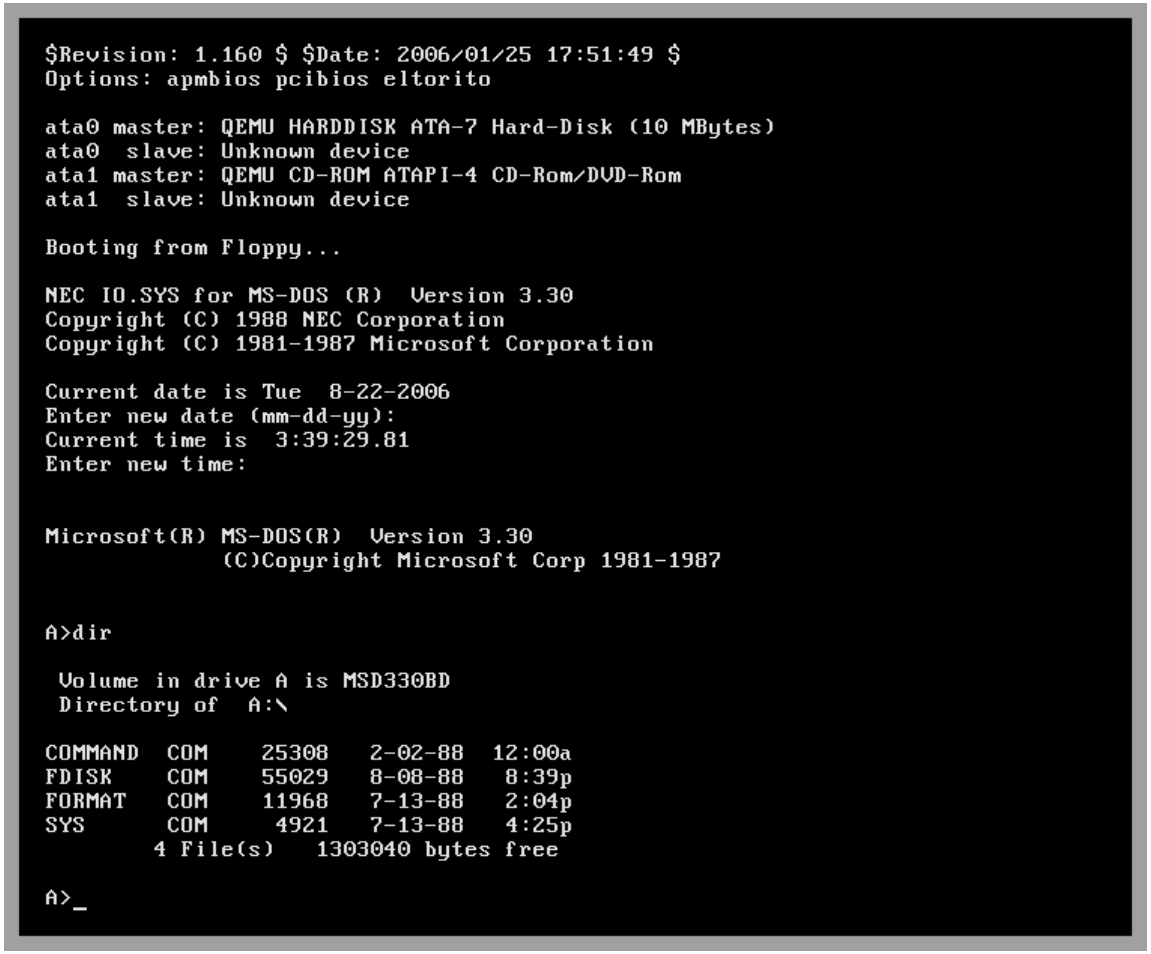
- Realno-časovni operacijski sistemi (real-time operating systems)
  - vgrajeni sistemi (embedded systems)
  - namenski računalniki
  - zagotavljajo zelo hiter odziv za kritične naloge
  - izvrševanje nalog po prioritetah

# Operacijski sistemi

- Microsoft DOS
- Microsoft Windows
- Microsoft Windows CE
- Mac OS
- DOS
- OS/2
- Unix
- AIX
- IRIX
- Mac X
- Solaris
- Linux
  - Ubuntu, Fedora, Debian, Gentoo, Mandrake, Slackware, Redhat,…
- Symbian
- …

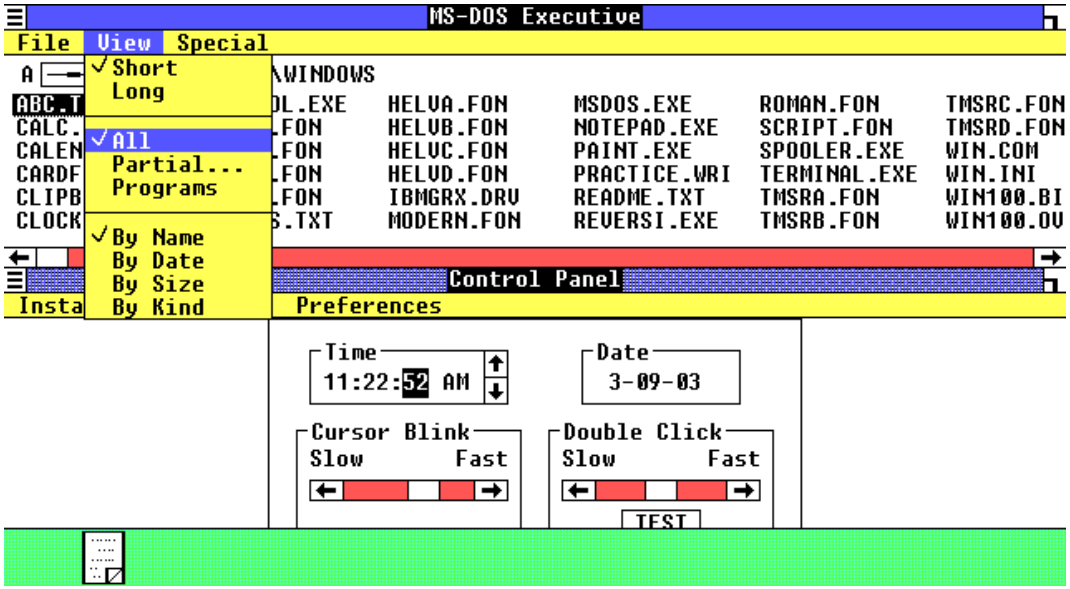
# MS DOS

- Microsoft
- Ukazna vrstica
- Brez GUI
- Od leta 1981
- Enouporabniški
- Enoopravilni
- Enoprocesorski
- Robusten



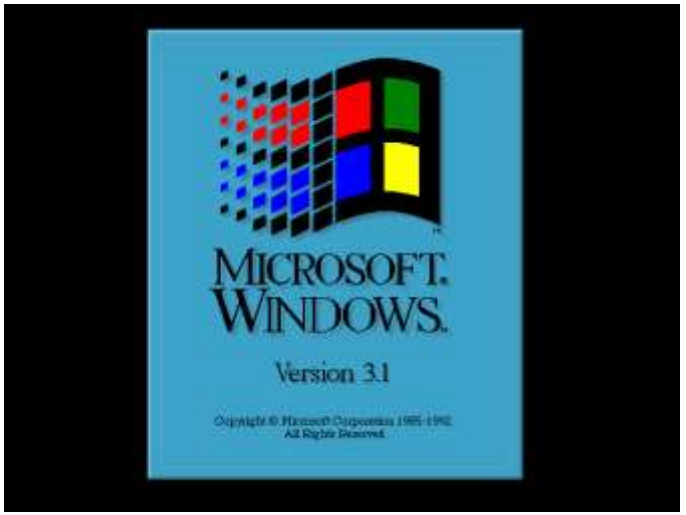
# MS Windows 1.0

- Microsoft
- Temeljil na MS DOS
- "Grafični" uporabniški vmesnik
- Od leta 1985
- Enouporabniški
- Enoopravilni
- Enoprocesorski

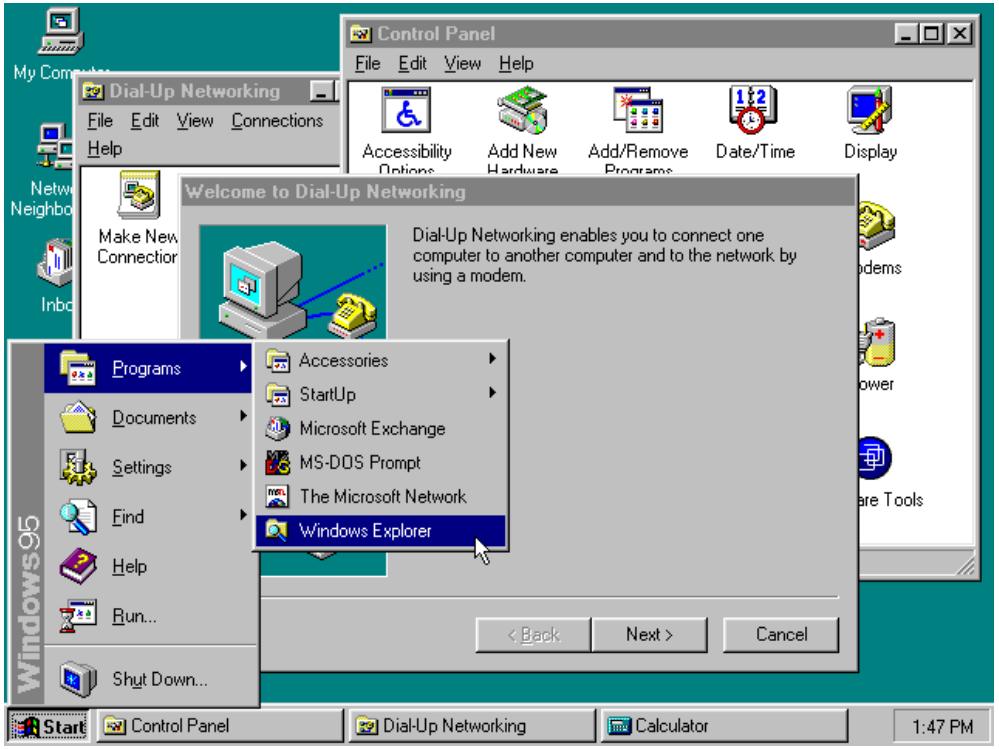
# MS Windows 3.1

- Microsoft
- Temeljil na MS DOS
- Grafični uporabniški vmesnik
- Od leta 1992-1995
- Nerobusten



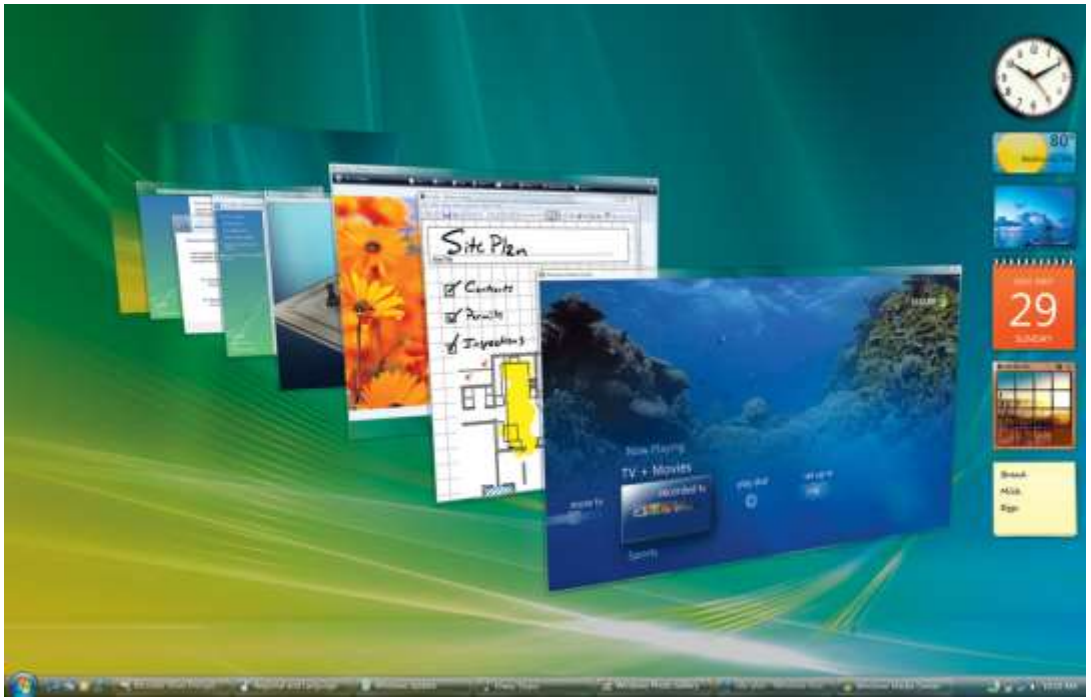

# MS Windows 95

- Microsoft
- Združil MS DOS in Windows
- Grafični uporabniški vmesnik
- Od leta 1995
- Večopravilen
- Robusten

# MS Windows Vista

- Microsoft
- Od 2007
- Prijazen uporabniški vmesnik
- Večuporabniški
- Veopravilni
- večprocesorski

# MS Windows 7

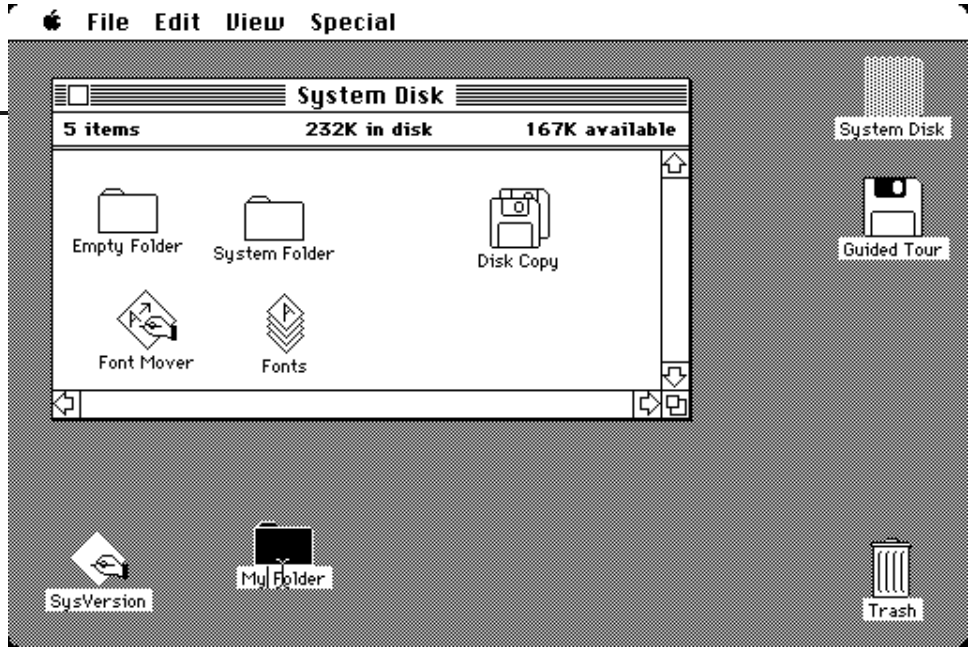
- Microsoft
- Uradno od oktobra 2009
- osnovni večdotični (multitouch) uporabniški vmesnik

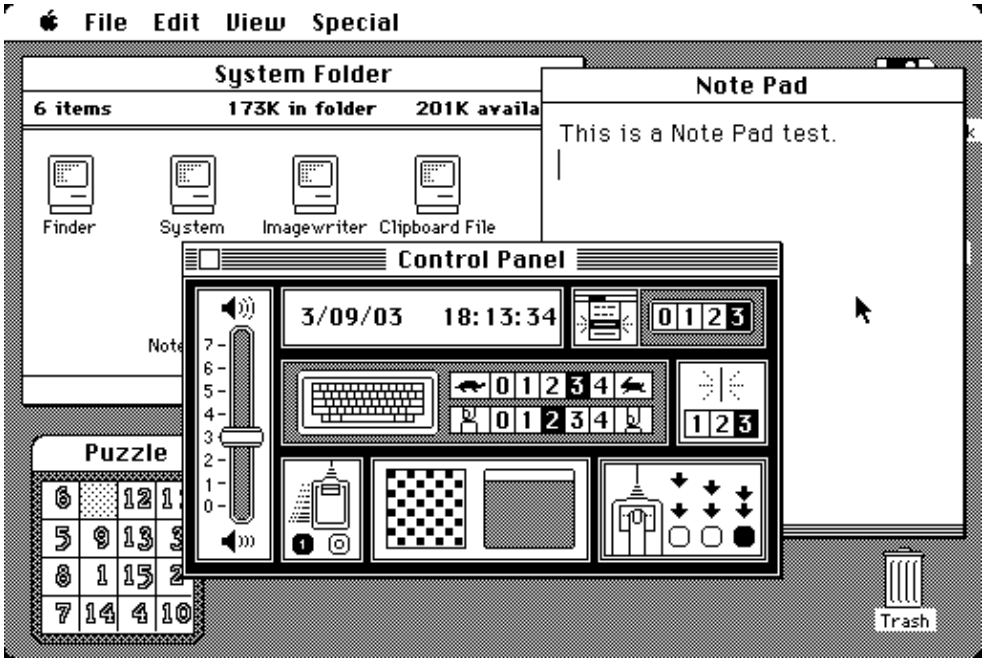

# MS Windows 8

- Microsoft
- Poudarek na večdotičnem uporabniškem vmesniku
- Usmerjenost tudi k večdotičnim napravam


# Mac OS System 1.0

- Apple Macintosh
- Prvi grafični uporabniški vmesnik
- 1984
- Miška



# Mac OS X 10.3 Panther

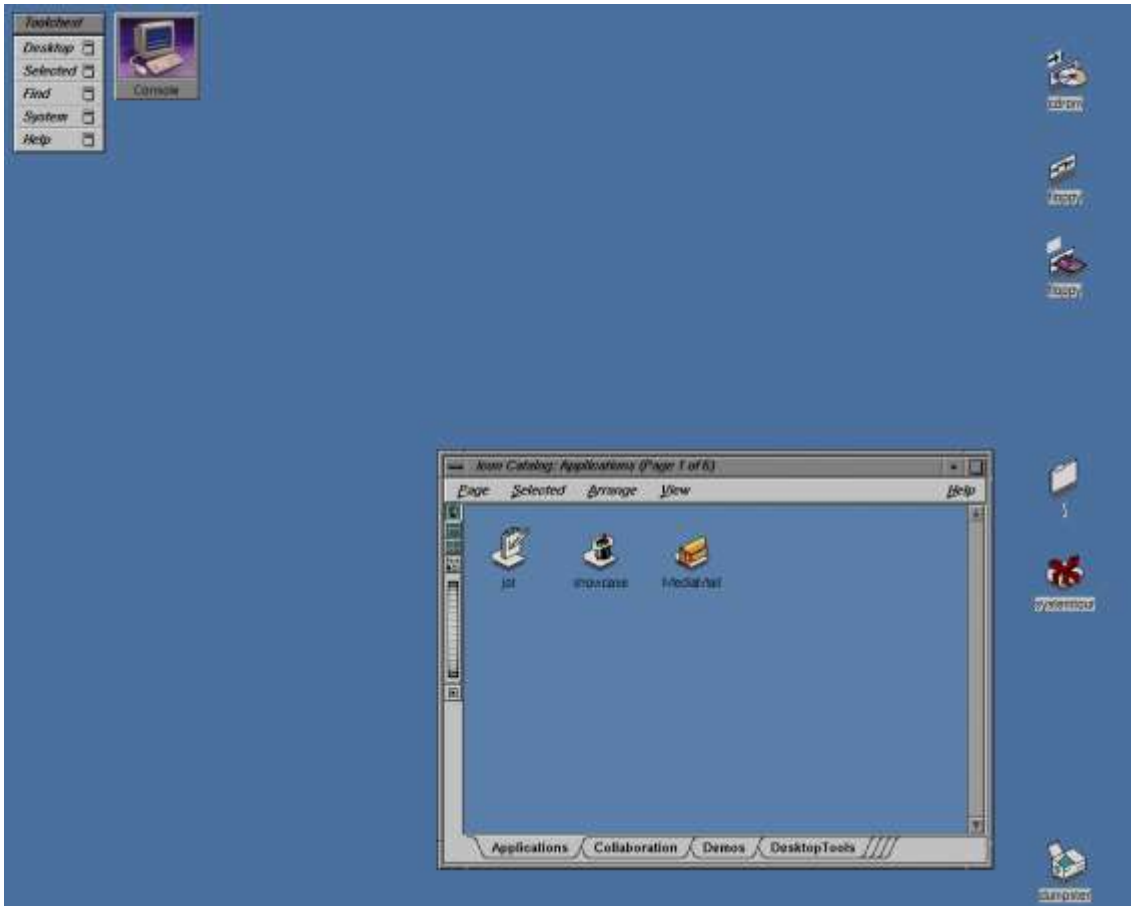
- Prijazen uporabniški vmesnik
- Zanesljivost delovanja

# Mac OS X Mavericks

- Povezljivost med vsemi napravami Apple

# IRIX 5.3.

- Silicon Graphics
- Unix
- 1994
- Irix 1.0: 1984

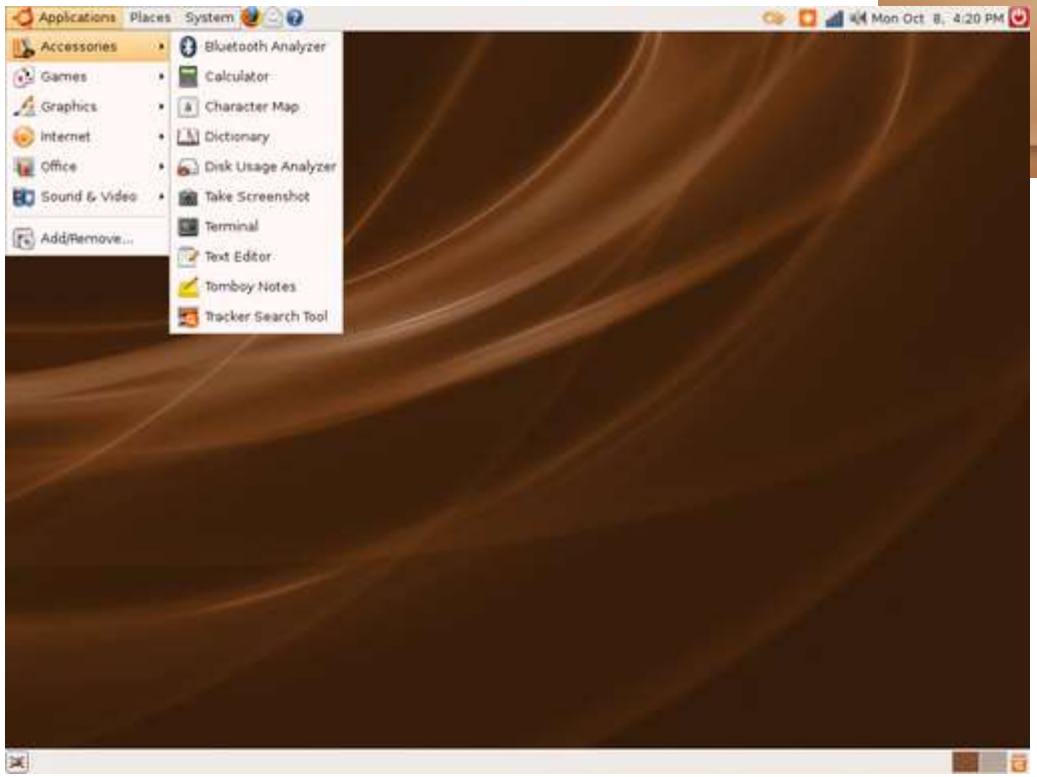


# Linux Redhat 9 in GNOME

- Linux Redhat 9
- Uporabniško okolje GNOME
- X windows
- Linux od 1991


# Linux Ubuntu

- Linux Ubuntu
- Zelo uporabniku prijazen
- Zelo razširjen
- Trenutna verzija: 13.10
- Brezplačen!


# OS za pametne mobilne naprave

- Android
- iOS
- Windows Mobile
- Symbian
- RIM BlackBerry OS



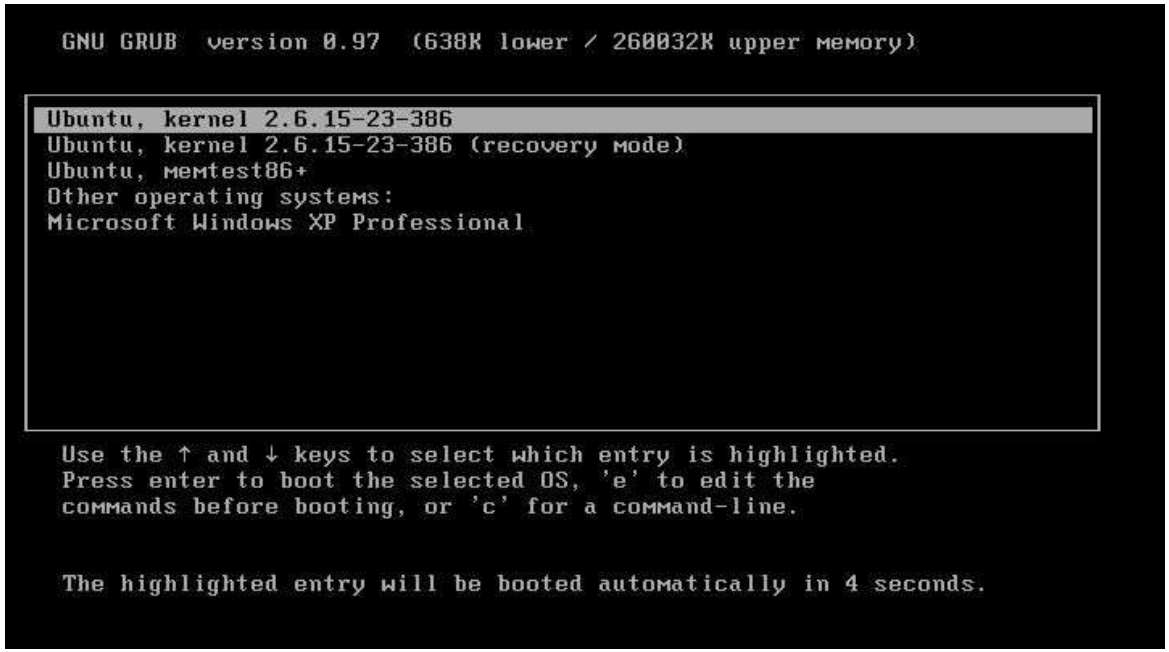







# Več operacijskih sistemov

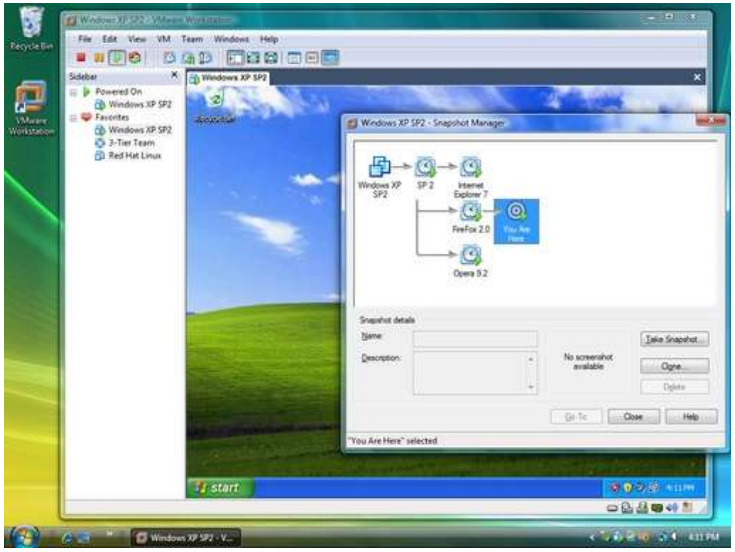
- Več operacijskih sistemov na enem računalniku
- Dual boot
  - Ločeno imamo nameščena dva operacijska sistema (ali več)
  - Poženemo samo enega naenkrat

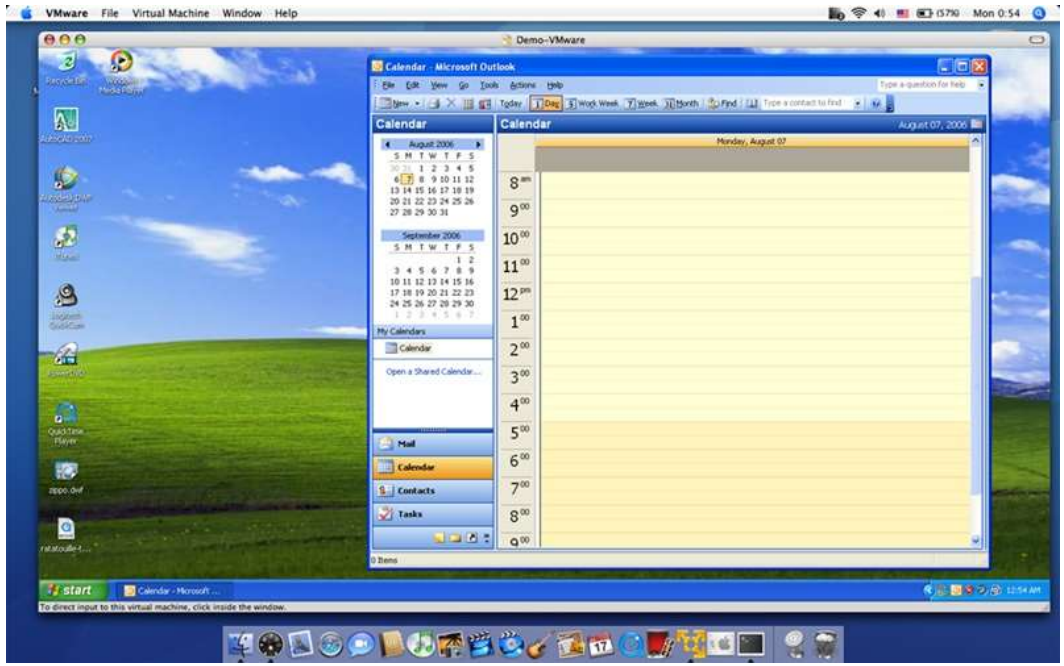



- Operacijski sistem lahko zaženemo tudi s CDja

# Virtualizacija

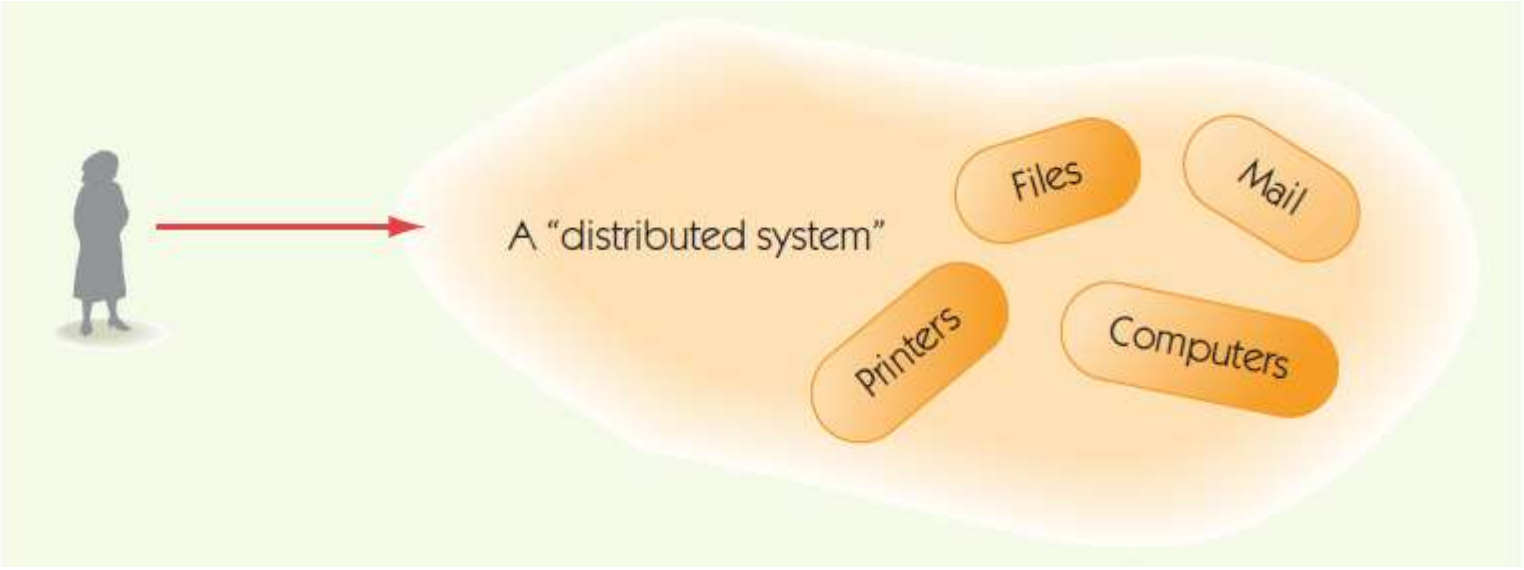
- Navidezni stroj
  - Poganjamo en operacijski sitem
  - V tem operacijskem sistemu poganjamo (emuliramo) druge operacijske sisteme
  - Lahko tudi izvajamo nekatere programe iz drugih operacijskih sistemov
  - Na enem fizičnem stroju lahko poganjamo več navideznih strojev





# Prihodnost

- Peta generacija: bližnja prihodnost
  - multimedijski uporabniški vmesniki
    - slike, govor, video, zvok, brezdotični vmesniki
  - vzporedni procesorski sistemi
    - enostavno vzporedno izvajanje programov
  - porazdeljeno računsko okolje
    - porazdeljeno izvajanje programov
    - pomembno je kaj naj se izvede, ne kje
    - računalništvo v oblaku



# Povzetek

- Sistemska programska oprema ustvari navidezni stroj, ki je enostaven za uporabo
- Zbirniki in nalagalniki so sistemska programska oprema
  - zbirnik prevaja človeku prijaznejše programe v strojni jezik
  - nalagalnik jih naloži v pomnilnik ter zažene
- Zbirni jezik uporablja simbolična imena, simbolične kode ukazov in pseudo operacije za opis algoritmov
- Operacijski sistem komunicira z uporabnikom s tekstovnim ali grafičnim uporabniškim vmesnikom
- Ključne naloge operacijskega sistema: uporabniški vmesnik, varnost sistema, razporejanje programov, varna uporaba virov
- Razvoj operacijskih sistemov skozi štiri generacije
- Prihodnost operaciskih sistemov: multimedija, paralelnost, porazdeljenost