

Uvod v računalništvo

Aleksander Sadikov

2015/2016

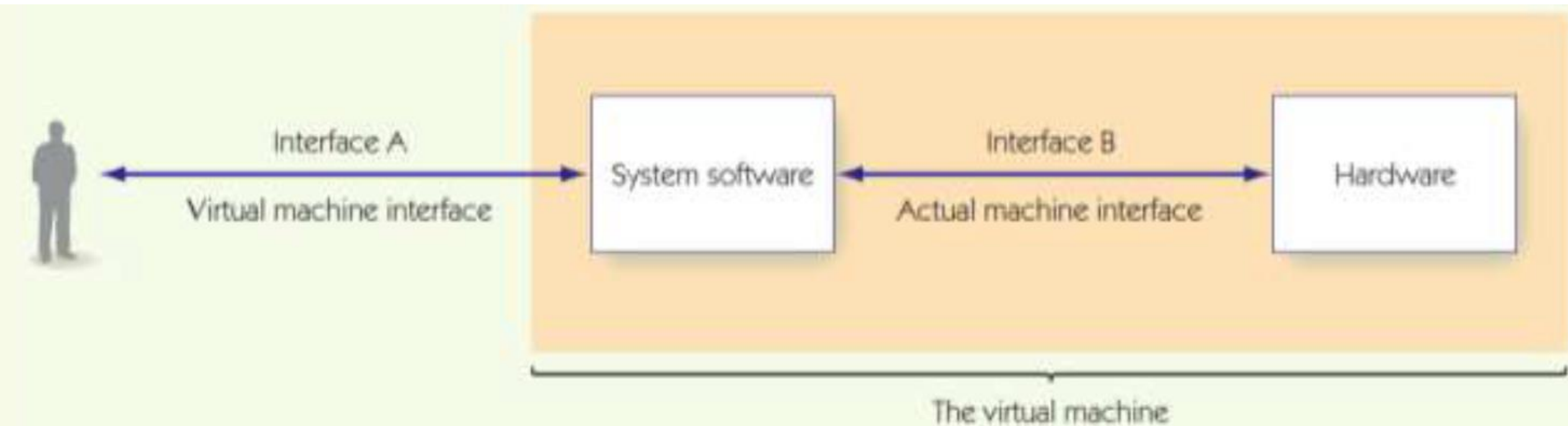
Naked machine.

01000101000010101010001010110

An interface is needed...

System software.

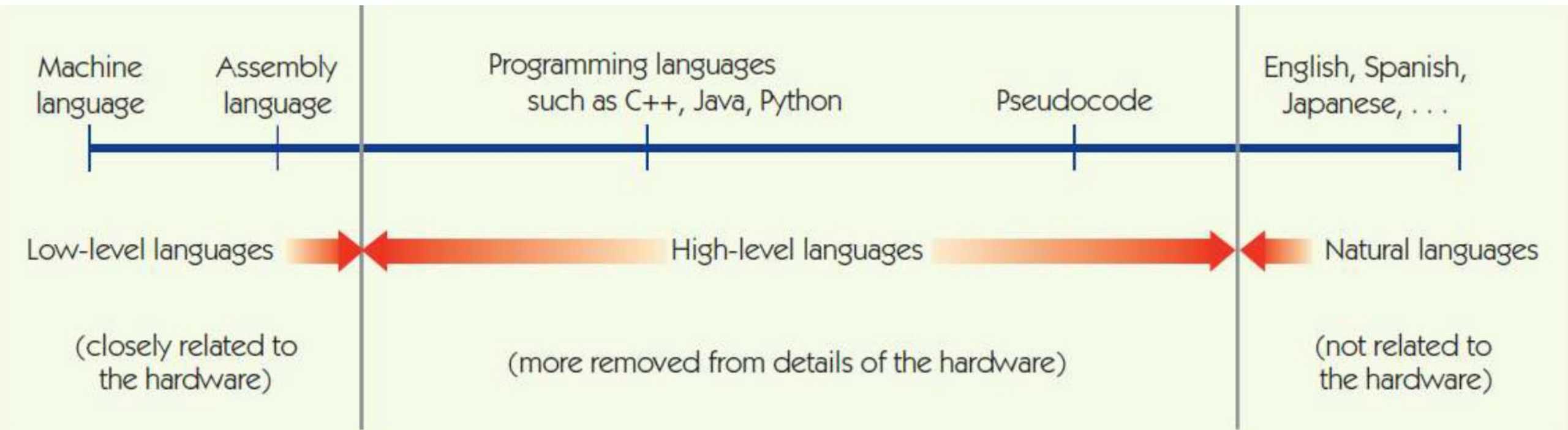
Virtual machine



Functions of the Operating System



Assembler and the assembly language



Binary Op Code	Operation	Meaning
0000	LOAD X	$CON(X) \rightarrow R$
0001	STORE X	$R \rightarrow CON(X)$
0010	CLEAR X	$0 \rightarrow CON(X)$
0011	ADD X	$R + CON(X) \rightarrow R$
0100	INCREMENT X	$CON(X) + 1 \rightarrow CON(X)$
0101	SUBTRACT X	$R - CON(X) \rightarrow R$
0110	DECREMENT X	$CON(X) - 1 \rightarrow CON(X)$
0111	COMPARE X	if $CON(X) > R$ then $GT = 1$ else 0 if $CON(X) = R$ then $EQ = 1$ else 0 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.

Let's do some assembly language programming...

OISC

Subtract and branch if less or equal to zero

subleq a, b, c

$\text{Mem}[b] = \text{Mem}[b] - \text{Mem}[a]$

if ($\text{Mem}[b] \leq 0$) goto c

Unconditional JUMP

subleq Z, Z, c

Addition: ADD a, b

subleq a, Z

subleq Z, b

subleq Z, Z

Copy: MOV a, b

subleq b, b

subleq a, Z

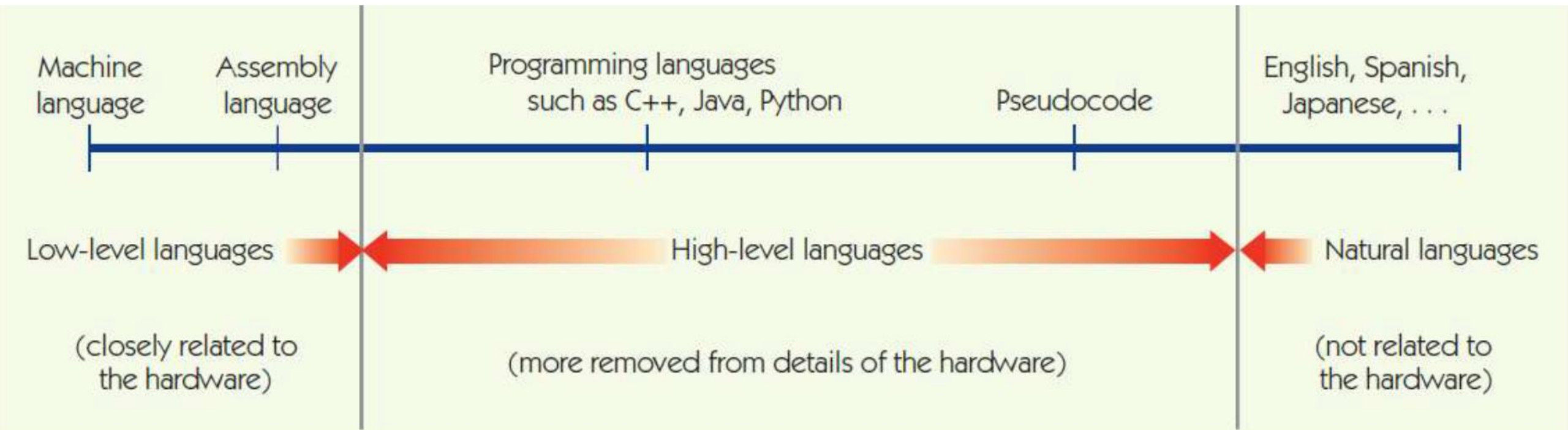
subleq Z, b

subleq Z, Z

branch-if-zero: BEQ b, c

```
        subleq b, Z, L1
        subleq Z, Z, OUT
L1:     subleq Z, Z
        subleq Z, b, c
OUT:    ...
```

Assembler and the assembly language



1. Convert symbolic op code to binary
2. Convert symbolic addresses to binary
3. Perform the assembler services (pseudo-ops)
4. Put the translated instructions into a file

Op code lookup table

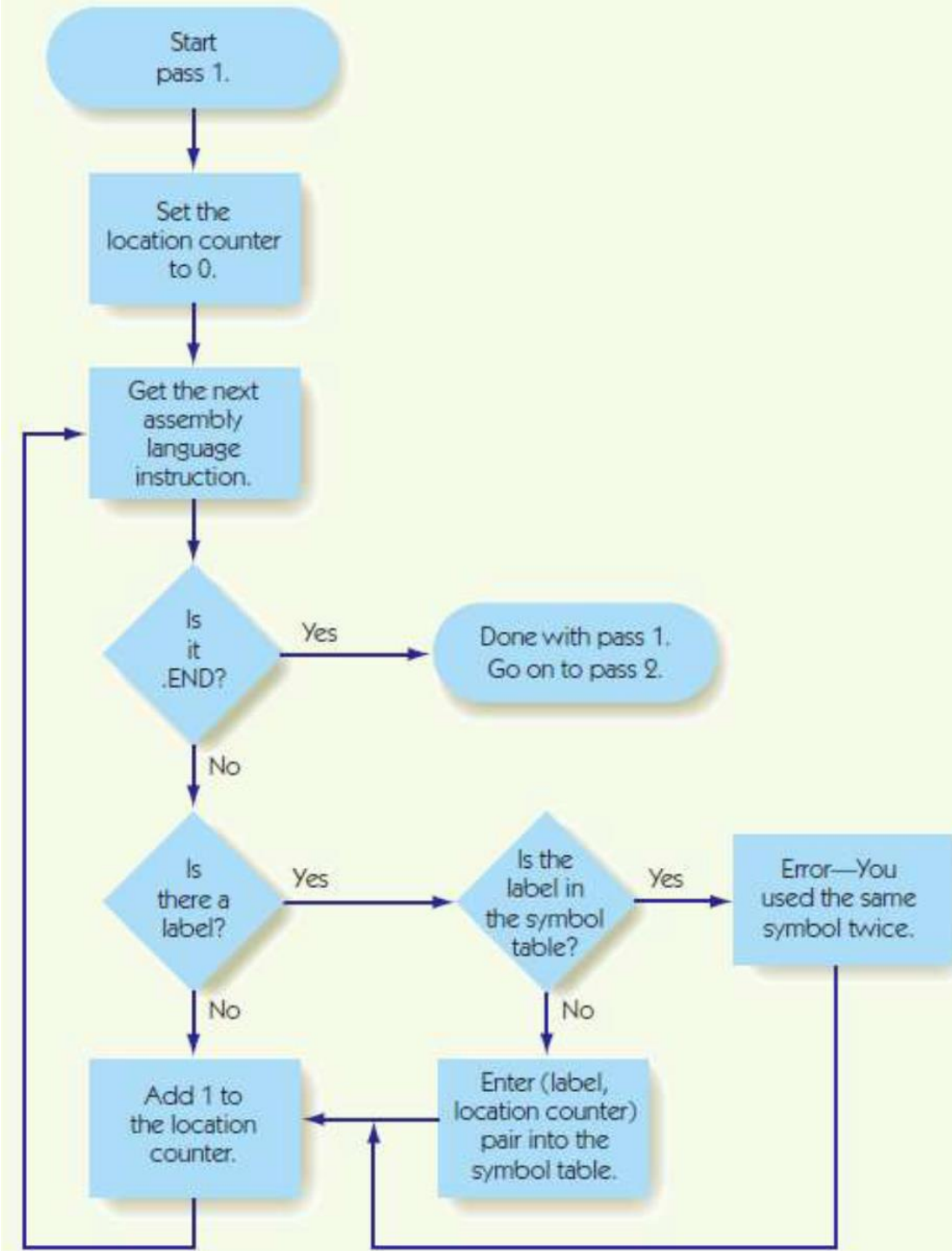
Binary Op Code	Operation	Meaning
0000	LOAD X	$CON(X) \rightarrow R$
0001	STORE X	$R \rightarrow CON(X)$
0010	CLEAR X	$0 \rightarrow CON(X)$
0011	ADD X	$R + CON(X) \rightarrow R$
0100	INCREMENT X	$CON(X) + 1 \rightarrow CON(X)$
0101	SUBTRACT X	$R - CON(X) \rightarrow R$
0110	DECREMENT X	$CON(X) - 1 \rightarrow CON(X)$
0111	COMPARE X	if $CON(X) > R$ then $GT = 1$ else 0 if $CON(X) = R$ then $EQ = 1$ else 0 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.

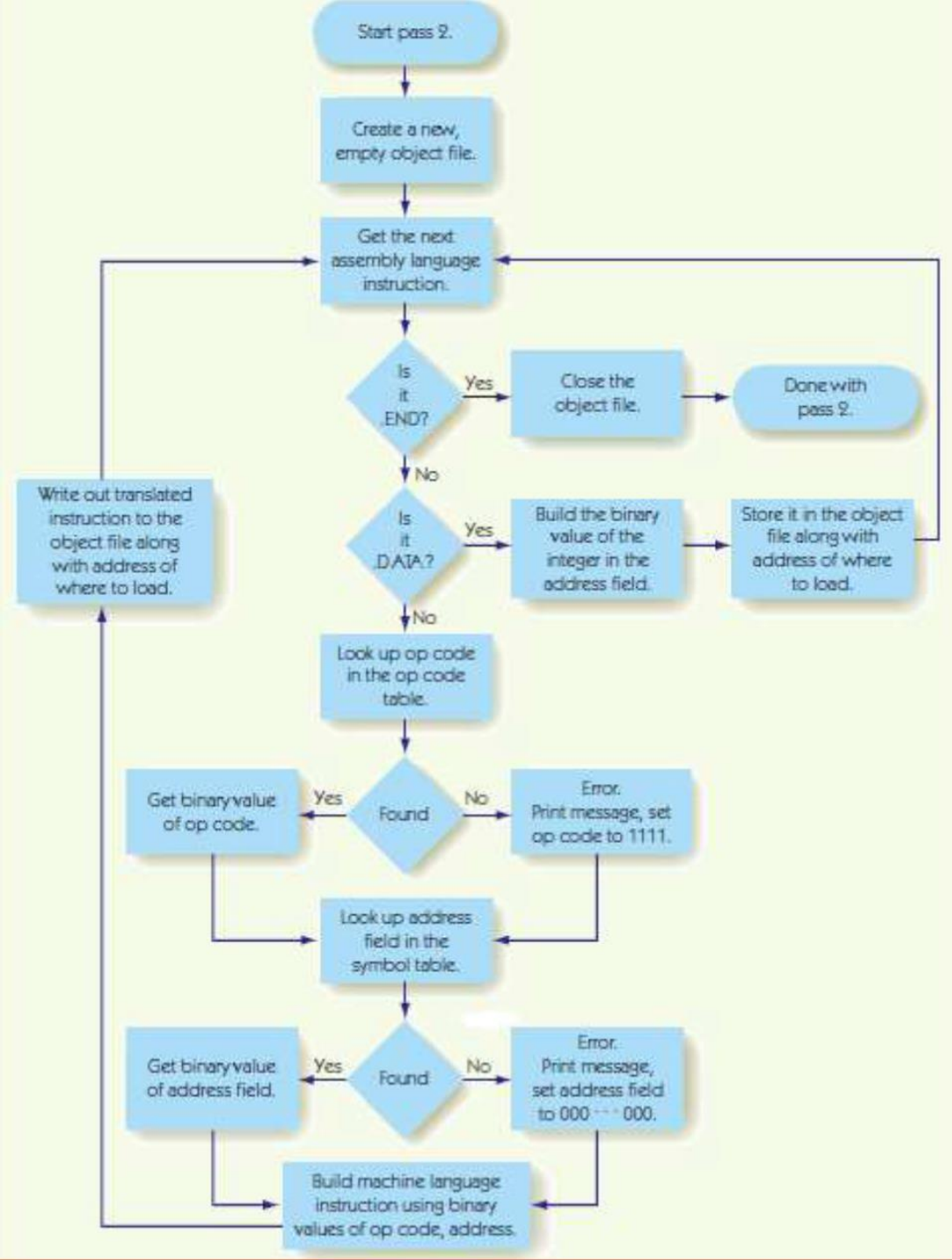
How about symbolic addresses?

(no lookup table exists for them)

Well, let's build one such lookup table ourselves...

PASS 1





PASS 2

1. Convert symbolic op code to binary
2. Convert symbolic addresses to binary
3. Perform the assembler services (pseudo-ops)
4. Put the translated instructions into a file

Let's translate...

So, who handles requests such as those below?

```
>>> assemble MyProg
```

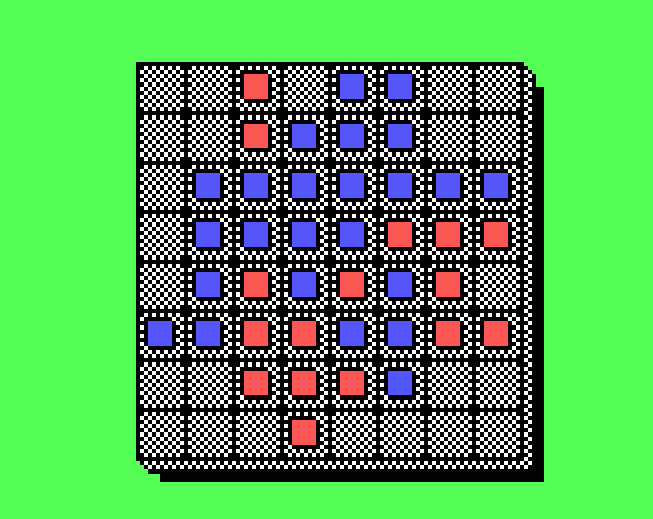
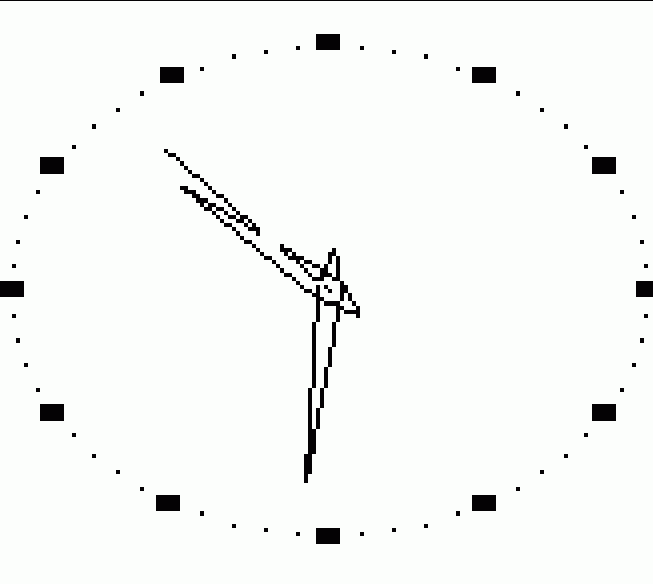
```
>>> run MyProg
```




November 20, 1985

Here we give an overview of some
of the (most important) functions of the OS.

Just some!





Microsoft Windows
MS-DOS Executive

Version 1.01

Copyright © 1985, Microsoft Corp.

Ok

Disk Space Free: 30024K
Memory Free: 303K

CONTROL.EXE	EGAMONO.GRB	HPLA
COURA.FON	EGAMONO.LGO	IBMC
COURB.FON	EMM.AT	JOYN
COURC.FON	EMM.PC	KERN

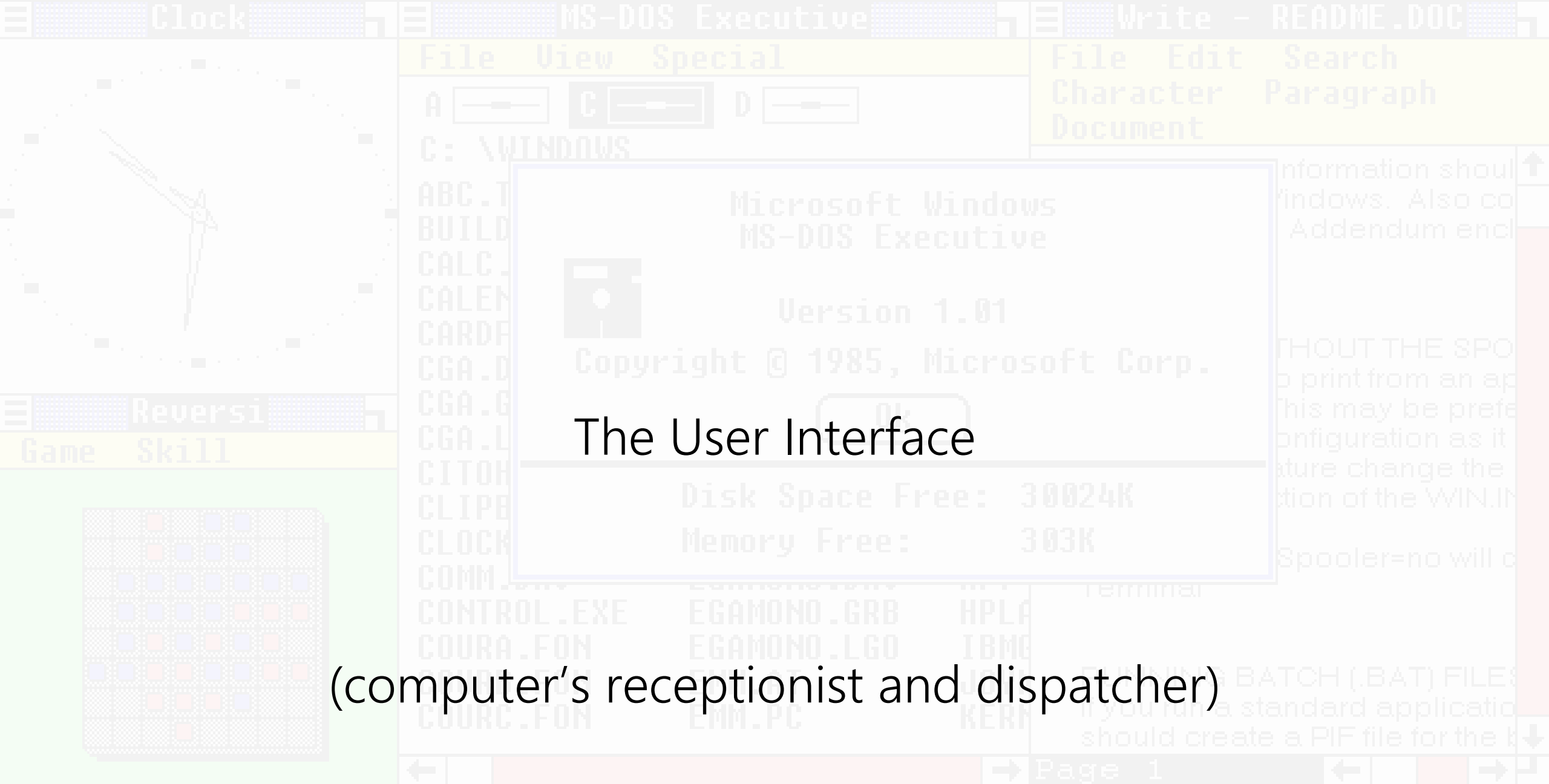
information should
Windows. Also co
Addendum encl

THOUT THE SPO
p print from an ap
This may be pref
onfiguration as it
ature change the
tion of the WIN.IN

Spooler=no will c

terminal

RUNNING BATCH (.BAT) FILES
If you run a standard application
should create a PIF file for the k



The User Interface

(computer's receptionist and dispatcher)

The command language

```
C:\> dir /s *.exe
```

```
C:\> copy con newScript.bat
```


System security and protection

(computer's security guard)

Responsibility of the OS:
to safeguard username/password combos

(and more)

Efficient allocation of resources

(computer's dispatcher)

Safe use of resources

(computer's traffic officer)

DEADLOCK

If a program cannot get all the resources that it needs,
it must give up all the resources it currently owns
and issue a completely new request.

Msg

Ack

Msg

Ack
(lost!)

deadlock

Msg (1)

Ack (1)

Msg (2)

Ack (2)
(lost!)

(wait 30secs)

Msg (2)

(discard duplicate copy)
Ack (2)

Msg (3)

The OS should create an illusion of a smoothly functioning, highly efficient, error-free environment – even, as we have seen, this is not always the case.

Take away lessons #P6

Cushions are important.
Cushions are very important, really.

Without cushions computers would **not**
penetrate every aspect of our lives.