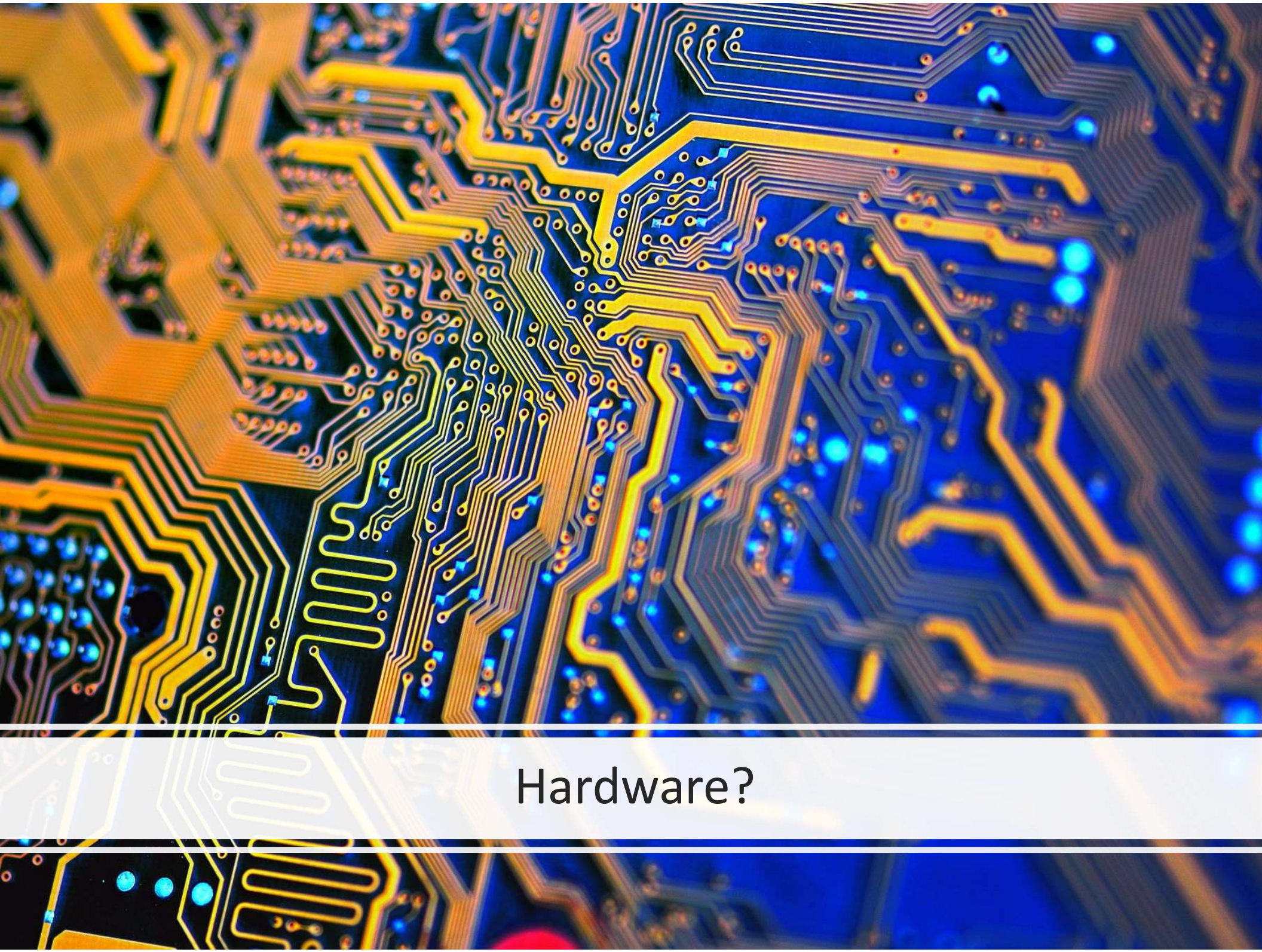
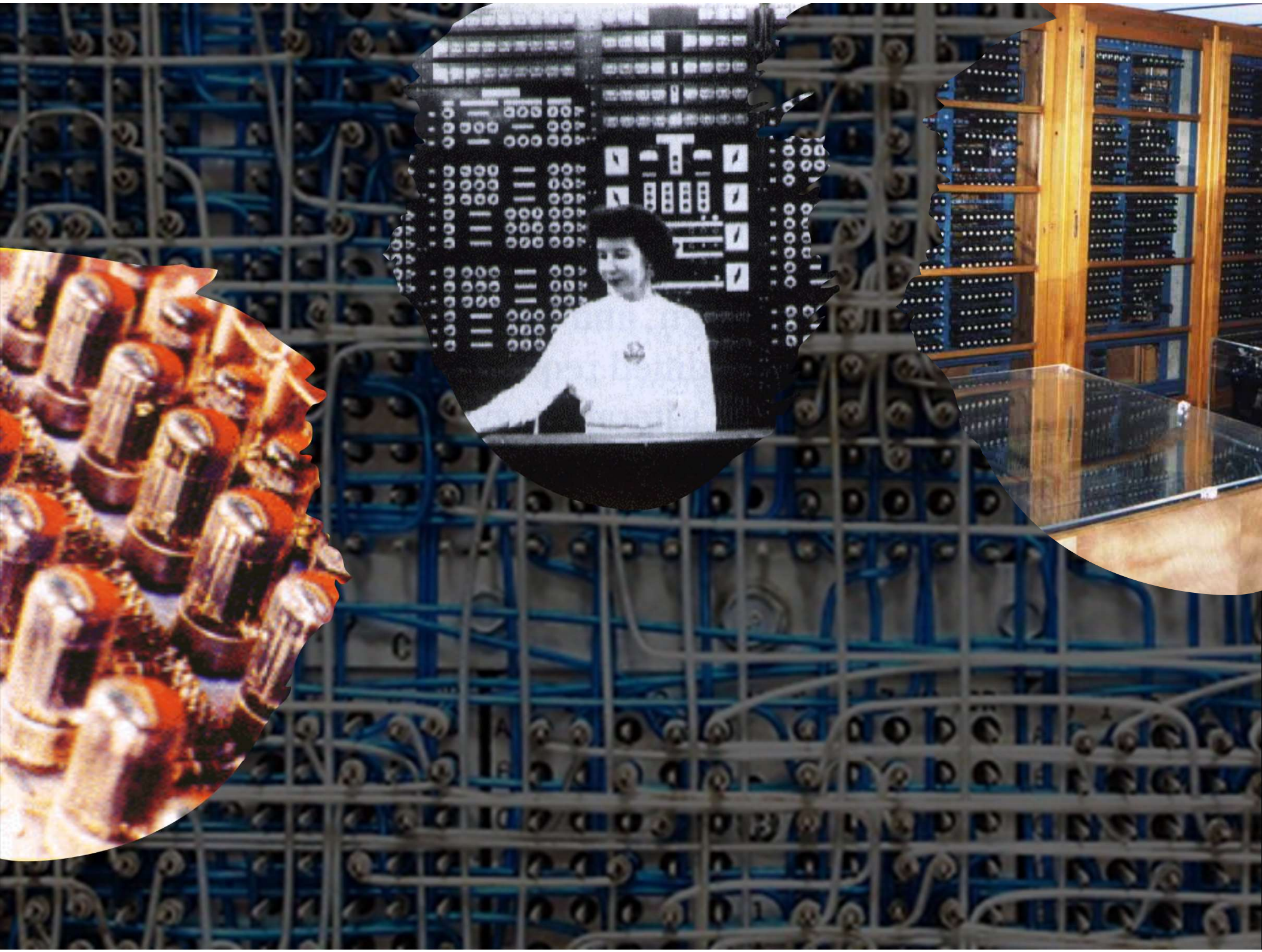
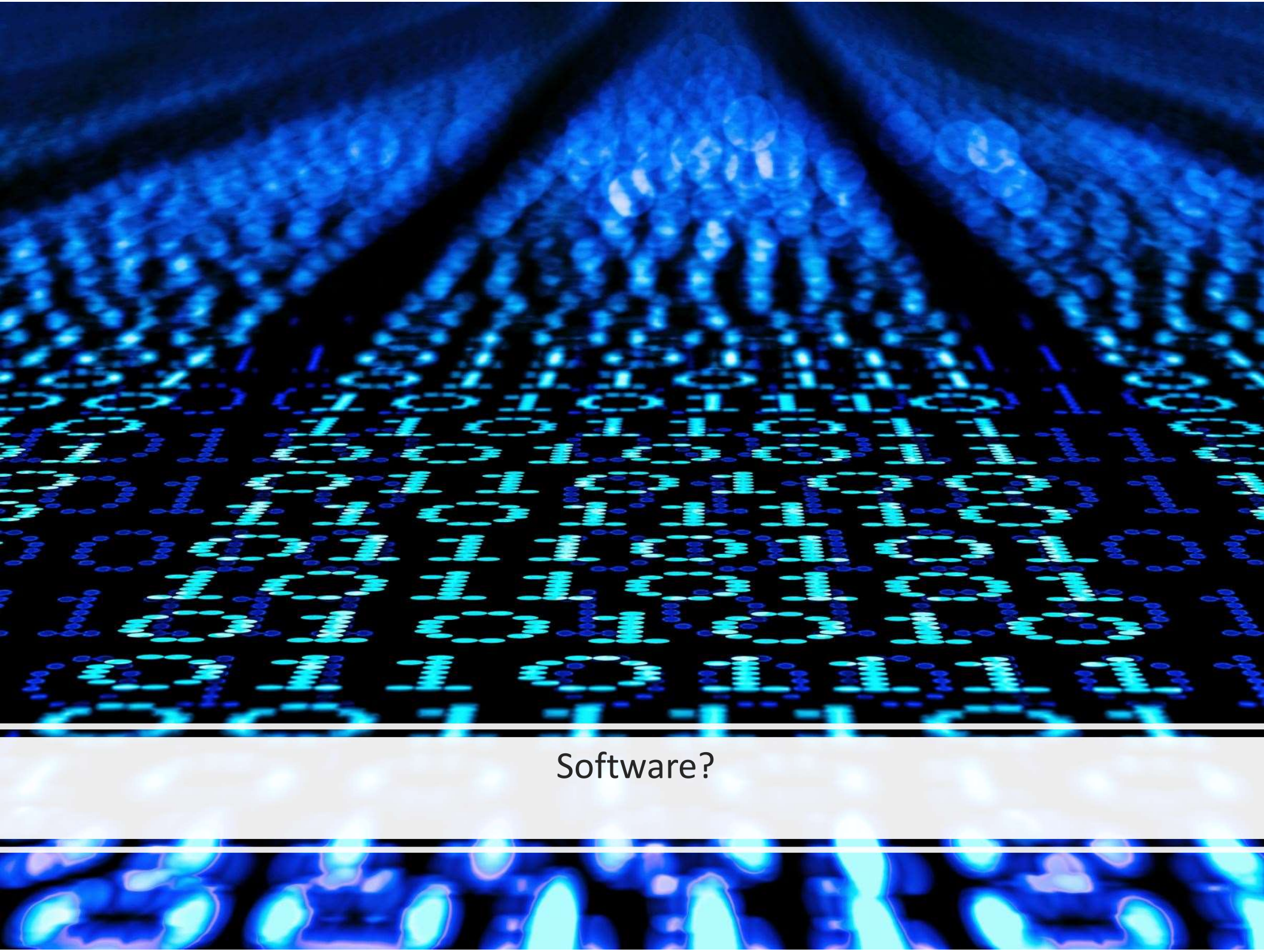
Why Computers?

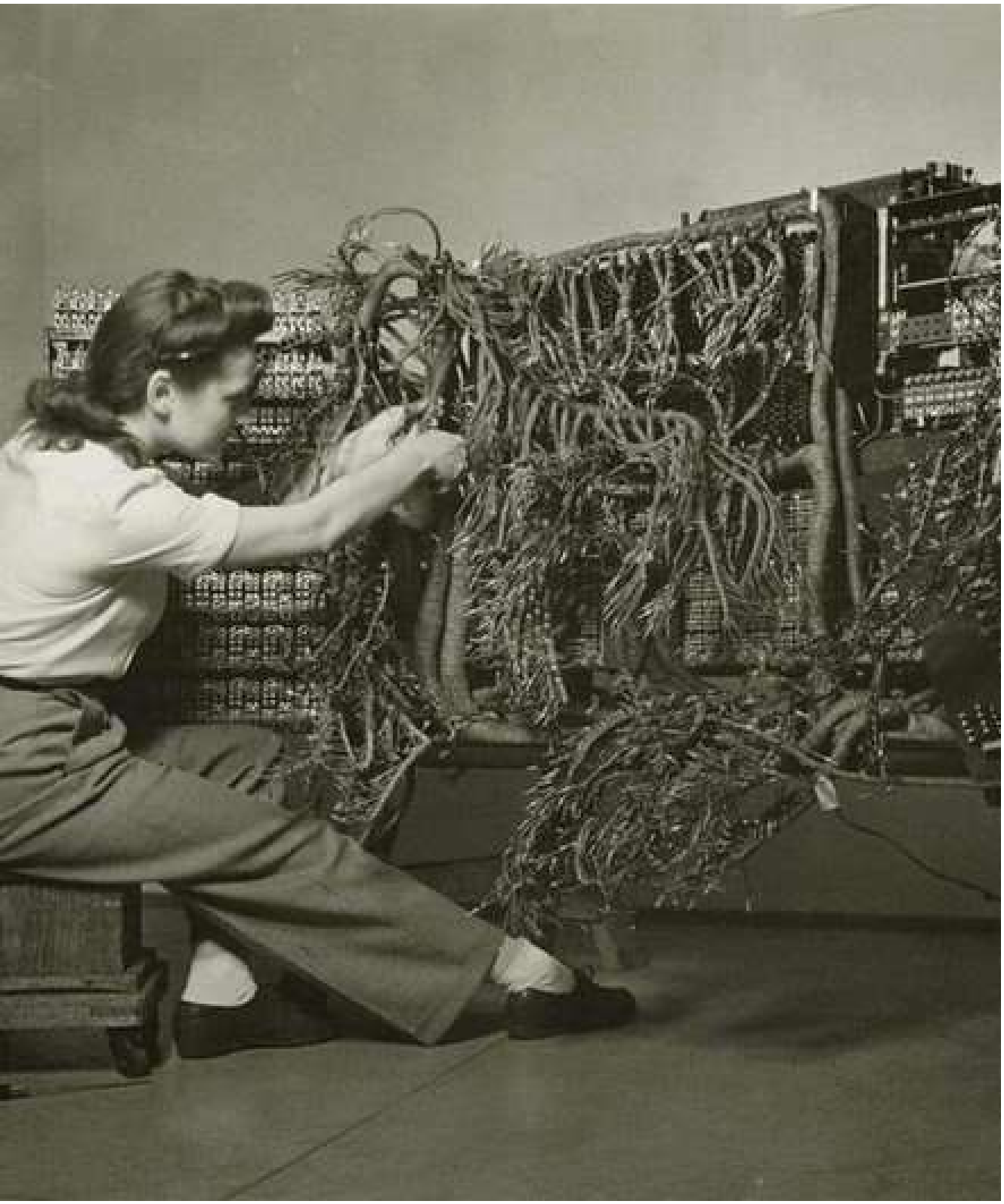


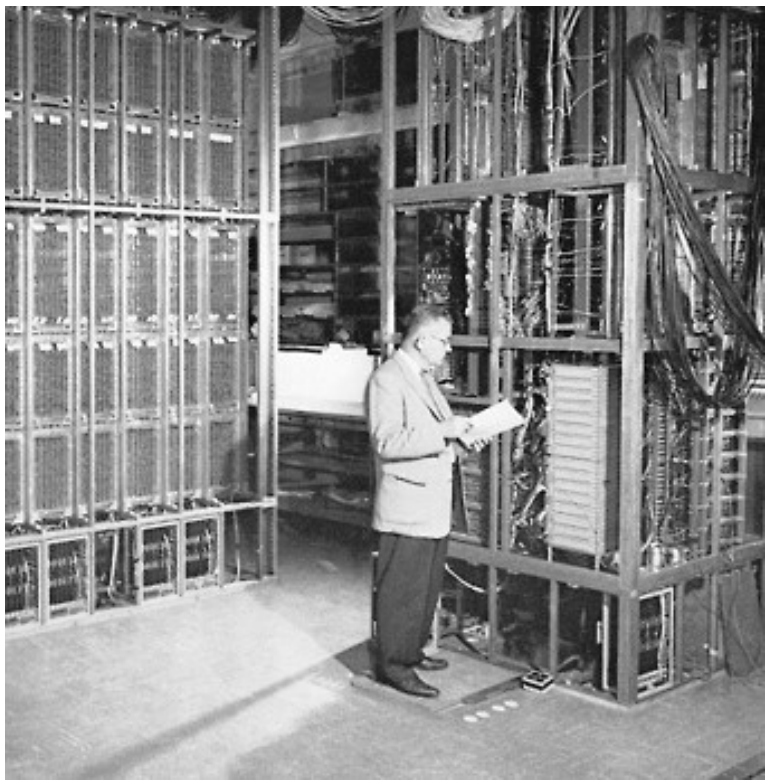
Hardware?





Software?

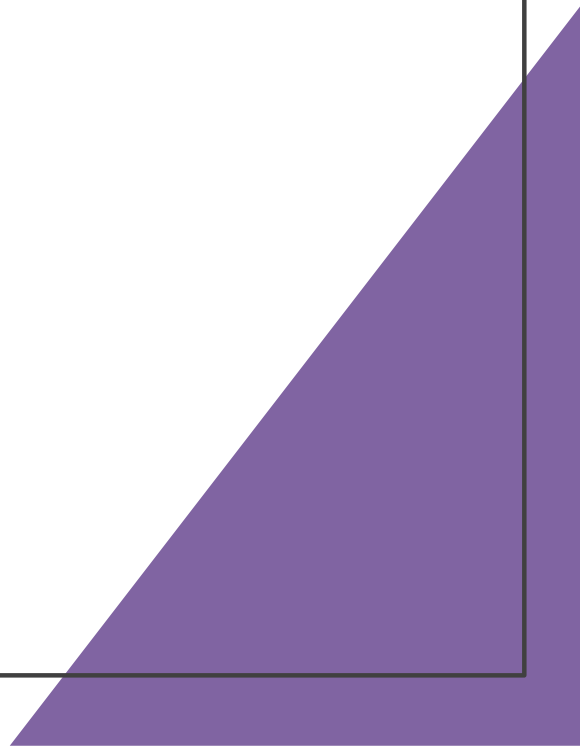


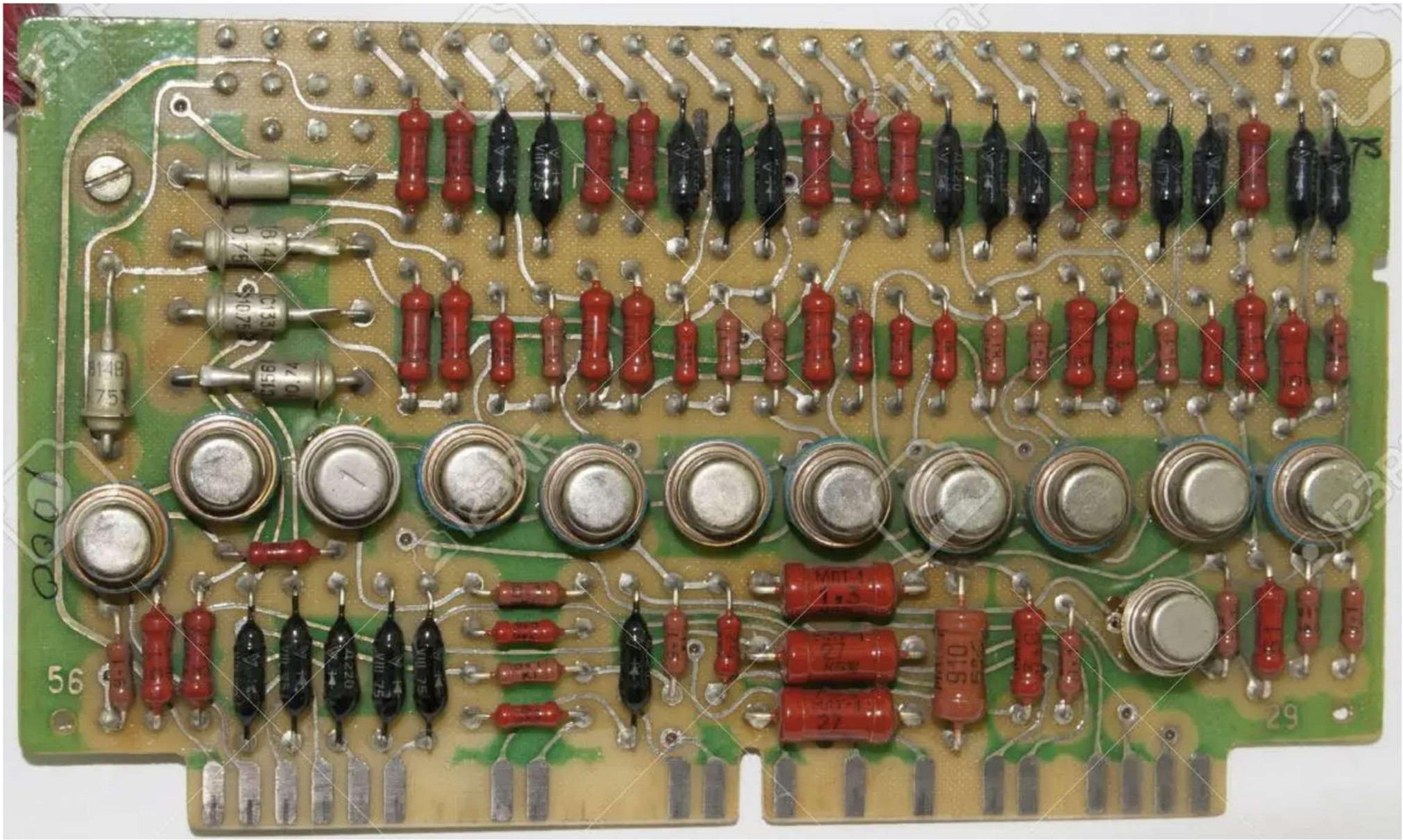


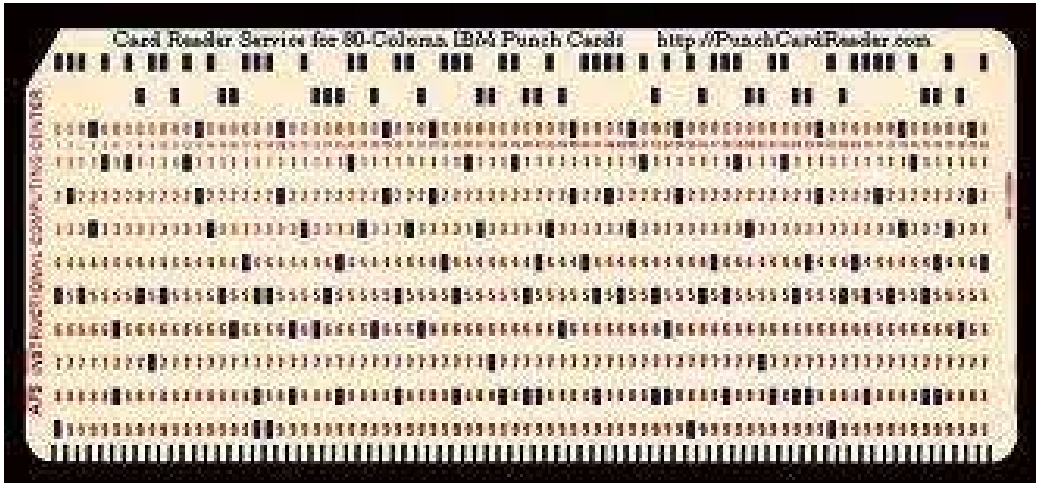

UNIVAC
Remington Rand
FIRST INSTRUCTION UNIT
SECOND INSTRUCTION UNIT
D.C. FAULT INDICATORS
STAND-BY HEATERS-LONG TANKS
HEATERS-SHORT TANKS

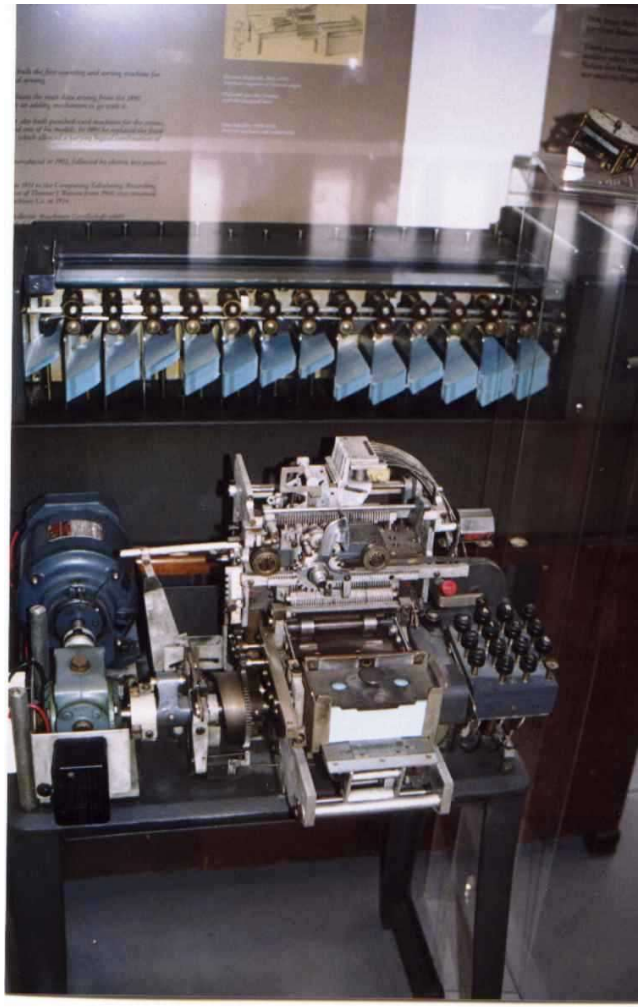
2nd Generation

- Hardware?

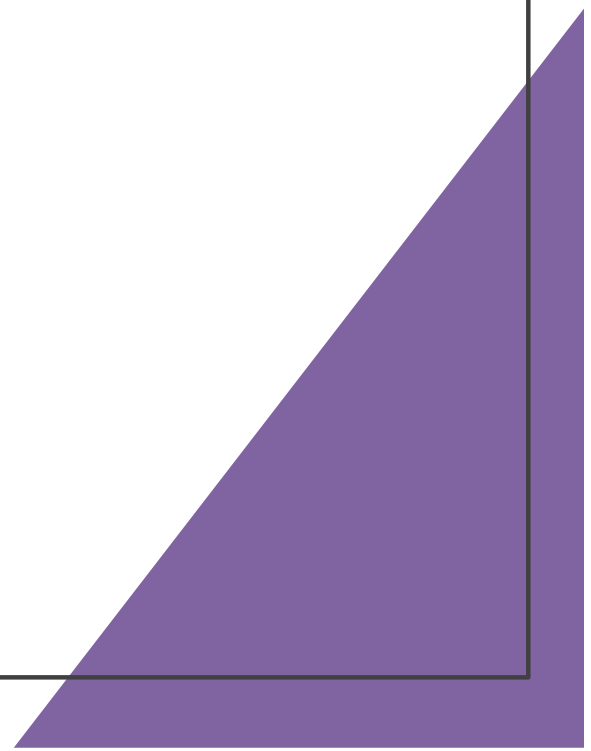





Card Reader Service for 80-Column IBM Punch Cards
http://PunchCardReader.com

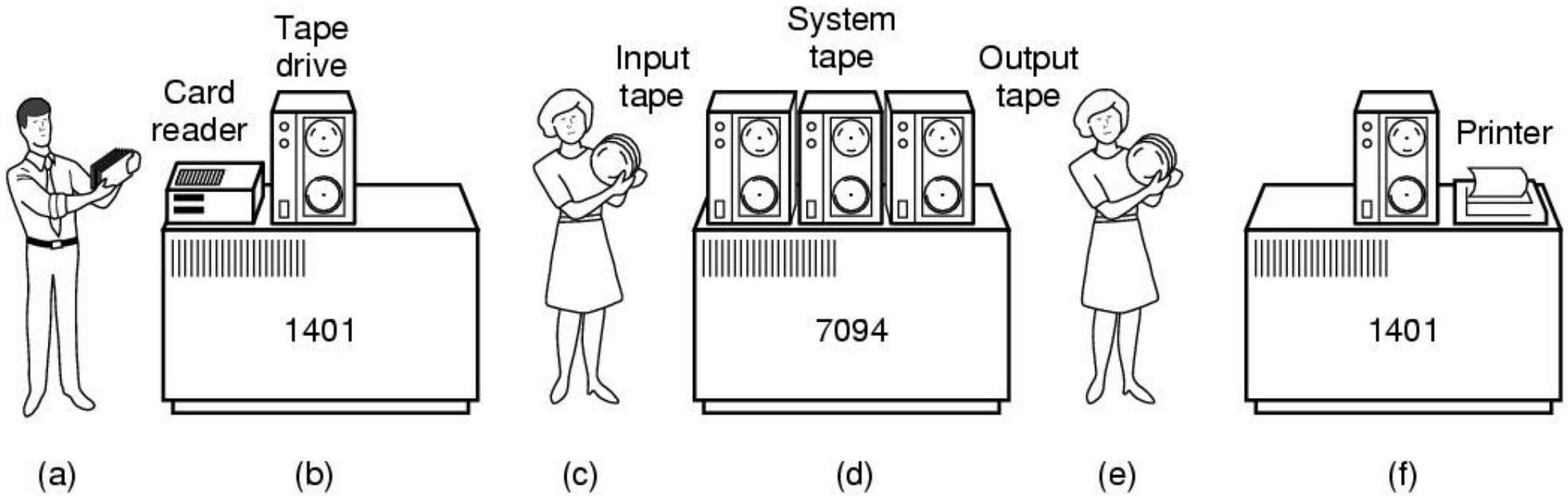


Software?



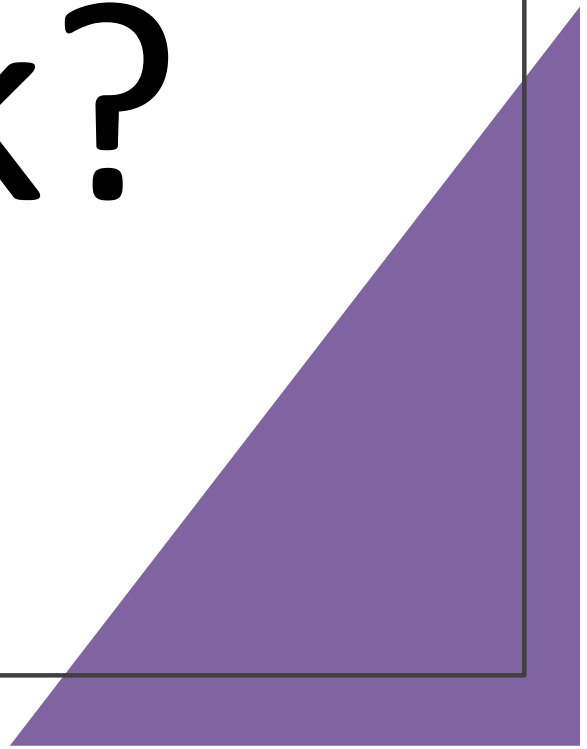
Bottleneck?


IBM
EXIT



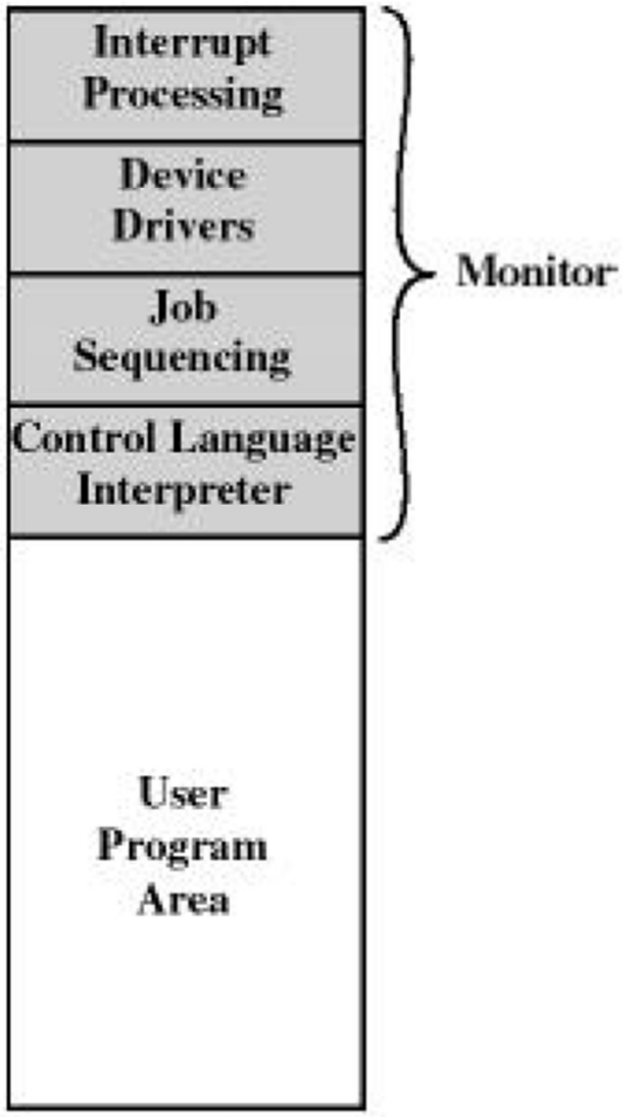
(a) Programmers bring cards to 1401.
(b)1401 reads batch of jobs onto tape.

Bottleneck?



The Monitor

- Monitor reads jobs one at a time from the input device
- Monitor places a job in the user program area
- A monitor instruction branches to the start of the user program
- Execution of user pgm continues until:
  - end-of-pgm occurs
  - error occurs
- Causes the CPU to fetch its next instruction from Monitor



Memory Layout of Resident Monitor

Simple Batch Systems

- Are the first operating systems (mid-50s)
- The user submit a job (written on card or tape) to a computer operator
- The computer operator place a batch of several jobs on a input device
- A special program, the monitor, manages the execution of each program in the batch
- Resident monitor is in main memory and available for execution
- Monitor utilities are loaded when needed

I/O Devices

- Controller (interface, channel, IOP) runs a device
- Small processor: commands from OS and executes them
- Complex
  - "Read sector x on disk y".
  - Convert to (cylinder, sector, head)
  - Move arm to correct cylinder,
  - wait rotation under the head
  - read and store bits coming off the drive,
  - compute checksum, store bits as words in memory

More bottlenecks

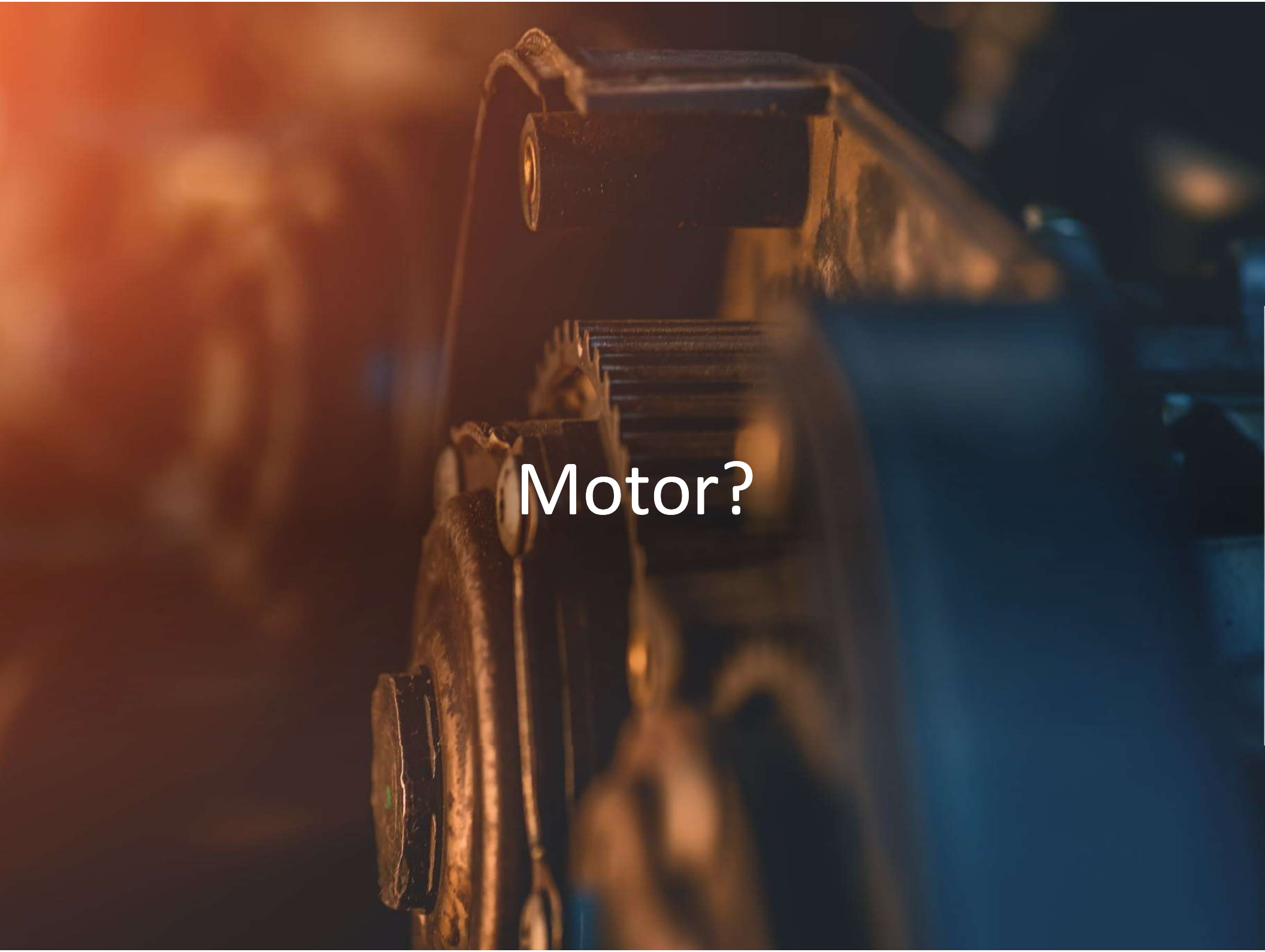
```
x, y: arrays of 80 chars each

 Loop 100 times {
     x = read (card);     // ~1s
     y = calculate_derivative(x,t);  // 0.1s
     print(y, printer);  // ~1s
}
```

Read card?

```
repeat 80 times {
    ch = check_card_contacts();
    advance_motor();
    card[i++] = ch;
}
i = 0;
get_next_card();
```

Print a line?

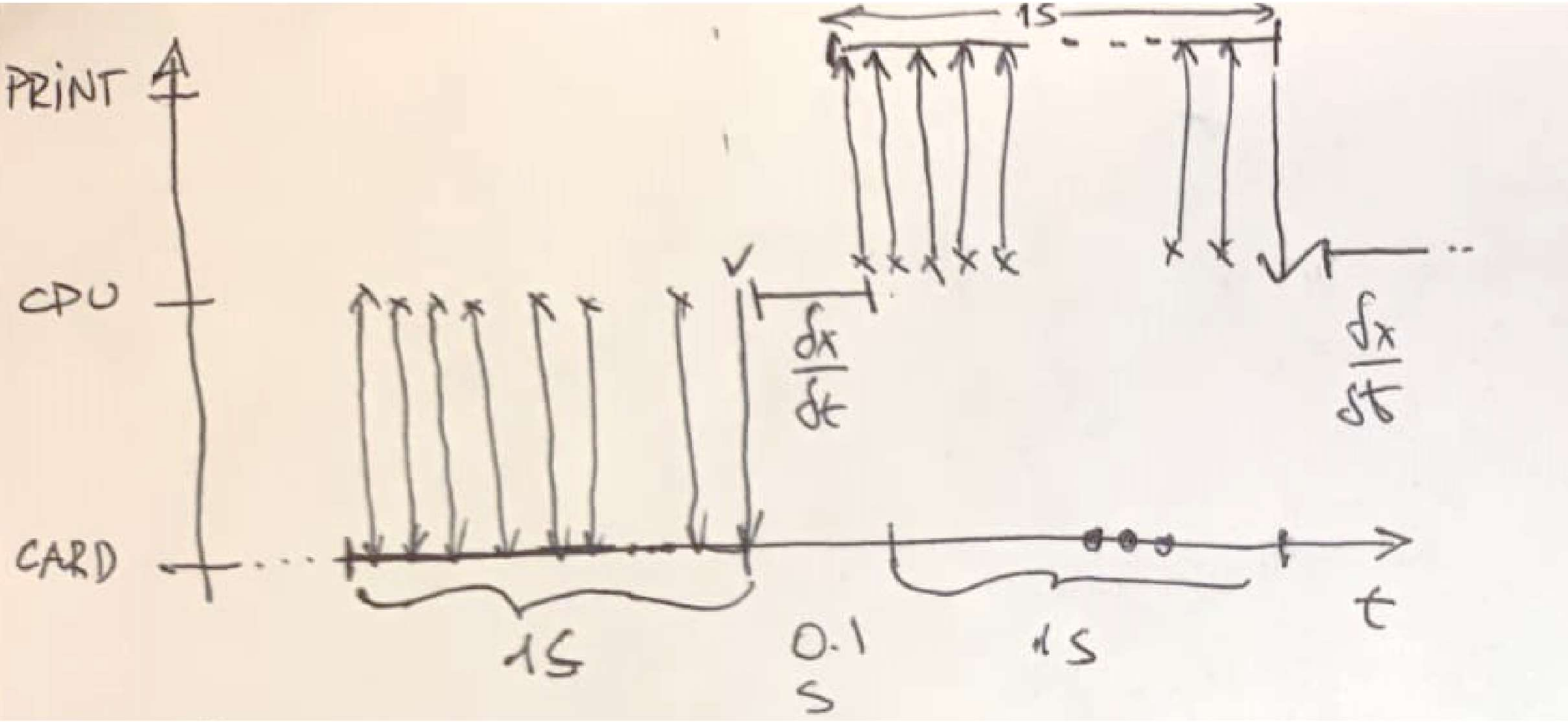
```
repeat 80 times {
    check_if_Printer_OK();
    print(output_text[i++]);
    advance_motor();
}
i = 0;
print(LF, CR);
```


Motor?

I/O by Timing

- Driver: command to controller
- Driver: waits a fixed amount of time
- Constant use of CPU
- Rarely used anymore

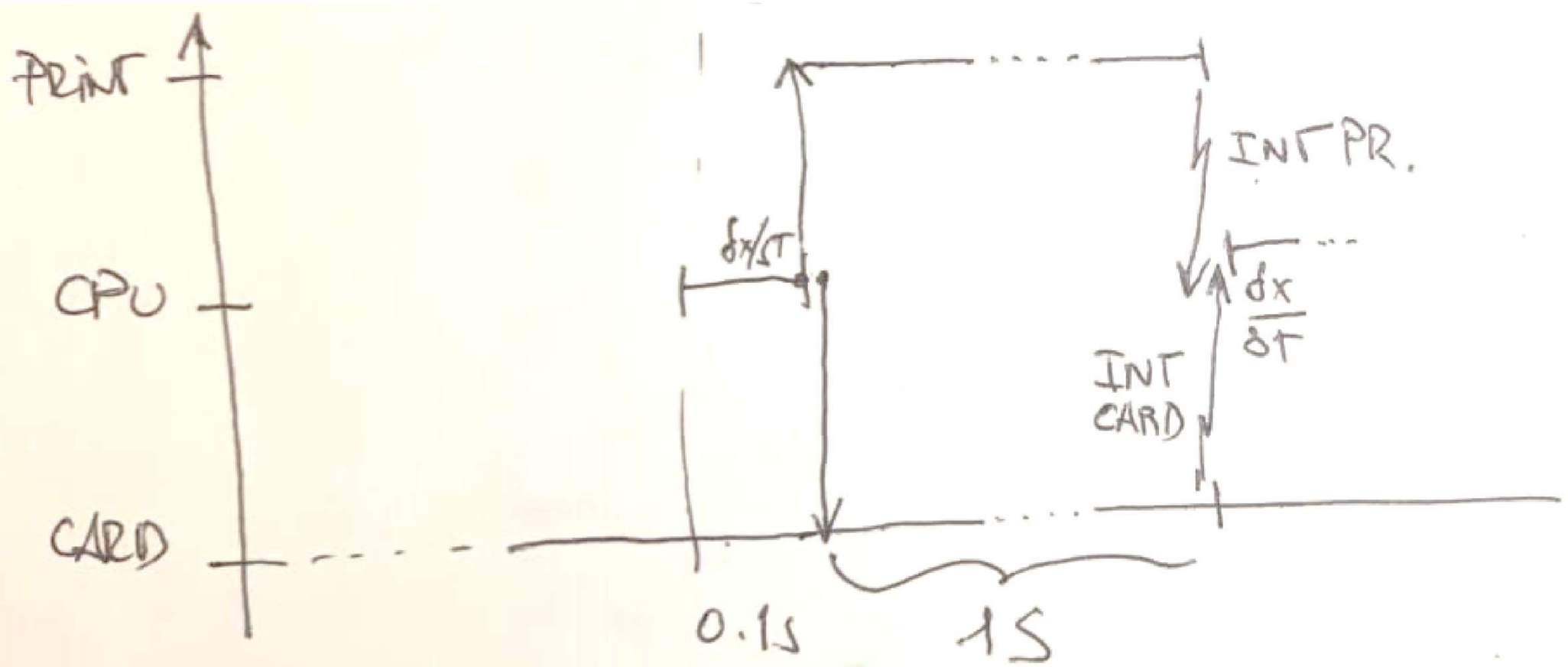
- Advantages?
- Disadvantages?


PRINT
CPU
CARD
15
$\frac{dx}{dt}$
$\frac{dx}{dt}$
15
0.1 s
1 s
t

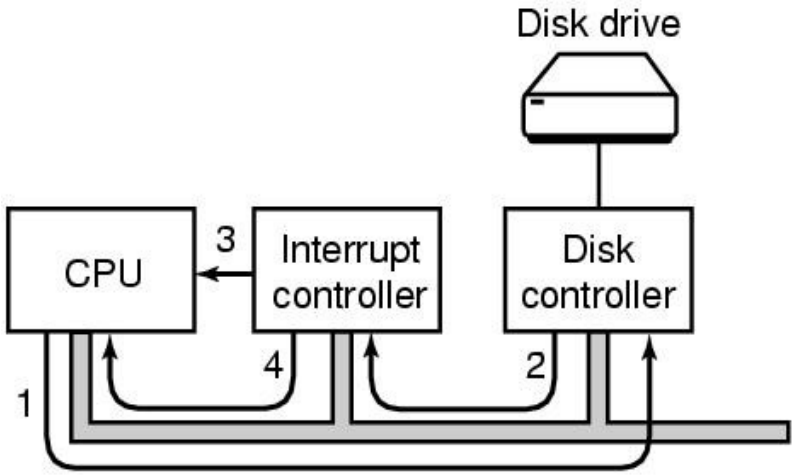
I/O by Polling

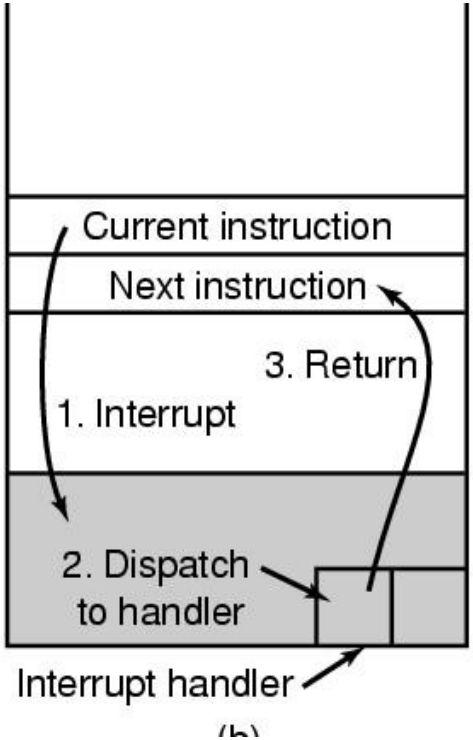
- Driver: command to controller
- Driver: polls device until it is ready
- Constant use of CPU
- Called programmed I/O
- rarely used any more

- Advantages?
- Disadvantages?


PRINT
CPU
CARD
INT PR.
INT CARD
0.1s
1s

I/O by Interrupts

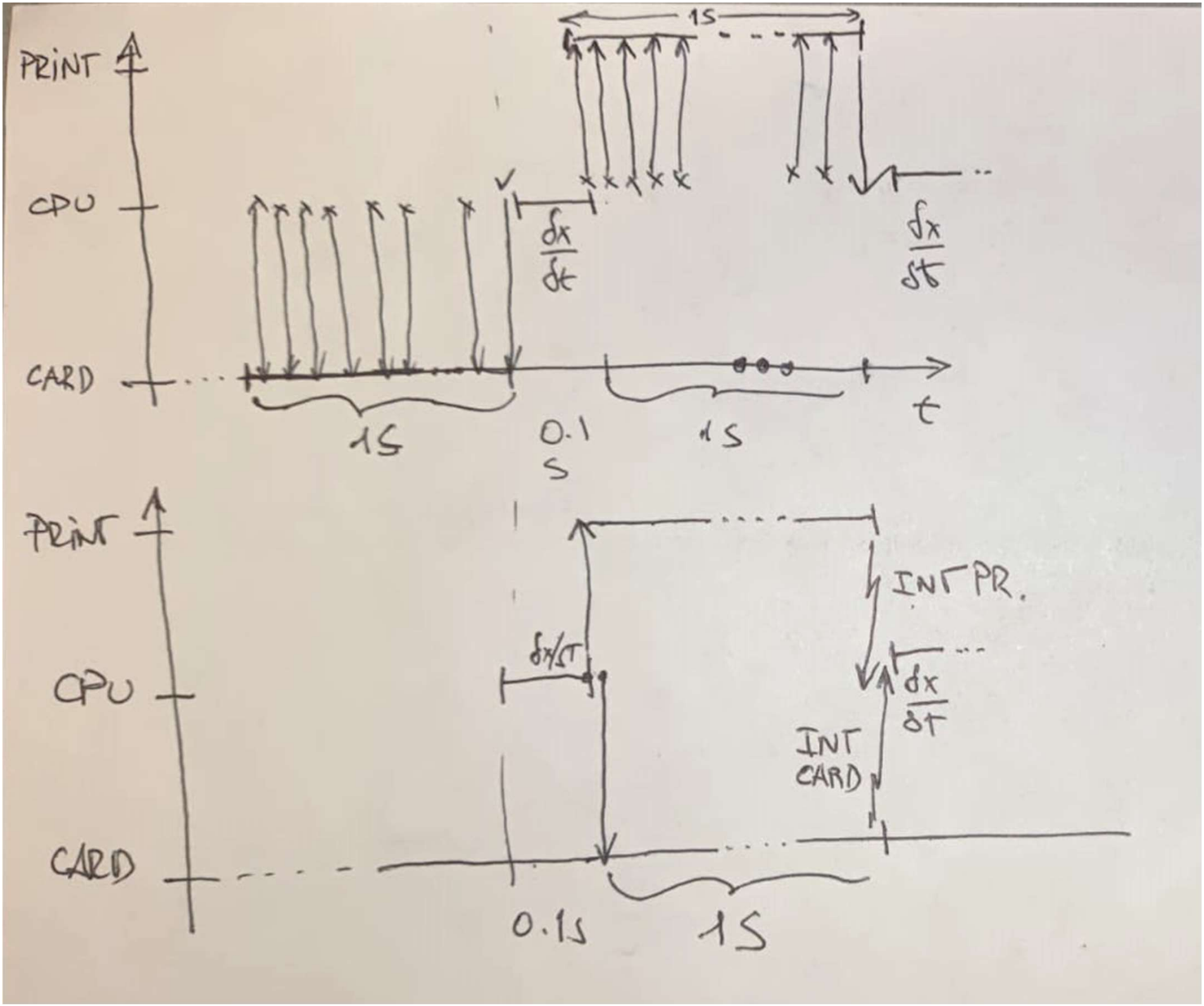




Generate an interrupt when I/O is finished.

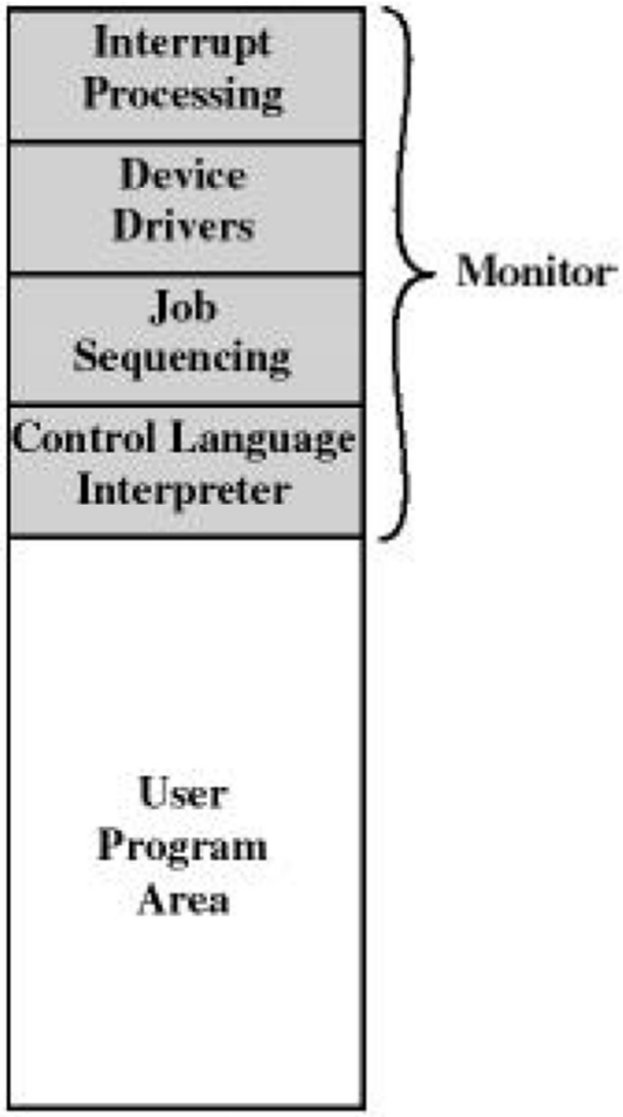
Eg When character is finished being printed, interrupt CPU.

Allows CPU to do something else while character is being printed

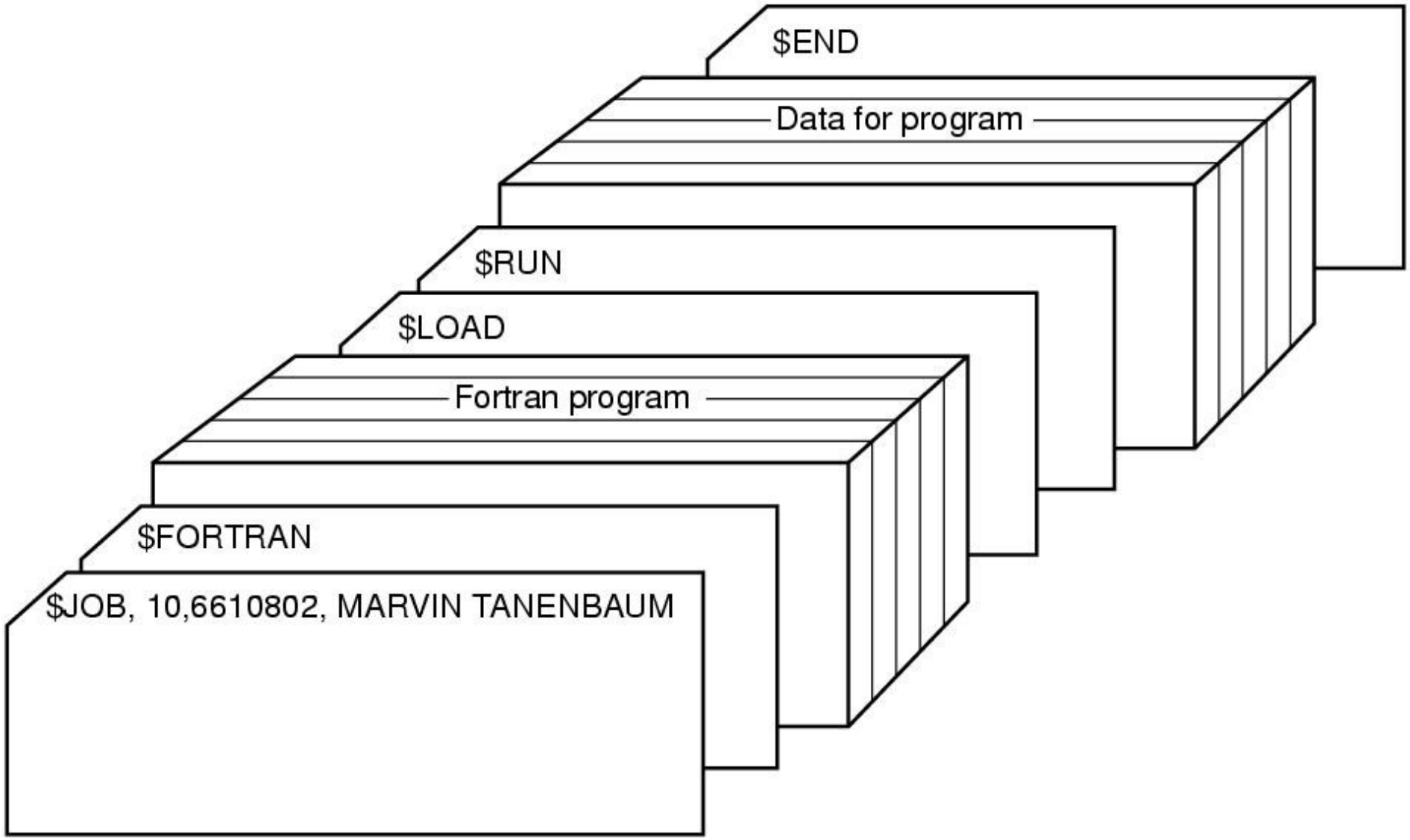

PRINT
CPU
CARD
1S
δx/δt
δx/δt
1S
0.1 S
1S
t
PRINT
CPU
CARD
δx/δt
INT PR.
δx/δt
INT CARD
0.1S
1S

The Monitor

- Monitor reads jobs one at a time from the input device
- Monitor places a job in the user program area
- A monitor instruction branches to the start of the user program
- Execution of user pgm continues until:
  - end-of-pgm occurs
  - error occurs
- Causes the CPU to fetch its next instruction from Monitor



Memory Layout of Resident Monitor


$END
Data for program
$RUN
$LOAD
Fortran program
$FORTRAN
$JOB, 10,6610802, MARVIN TANENBAUM

Job Control Language (JCL)

- Is the language to provide instructions to the monitor
  - what compiler to use
  - what data to use
- Example of job format: ------->>
- `$FTN` loads the compiler and transfers control to it
- `$LOAD` loads the object code (in place of compiler)
- `$RUN` transfers control to user program

```
$JOB
$FTN
. . .
FORTRAN
program
. . .
$LOAD
$RUN
. . .
Data
. . .
$END
```

Job Control Language (JCL)

- **Each read instruction (in user pgm) causes one line of input to be read**

- **Causes (OS) input routine to be invoke**
  - checks for not reading a JCL line
  - skip to the next JCL line at completion of user program