Organizacja i architektura komputerów

Asembler & GDB cd

Piotr Patronik

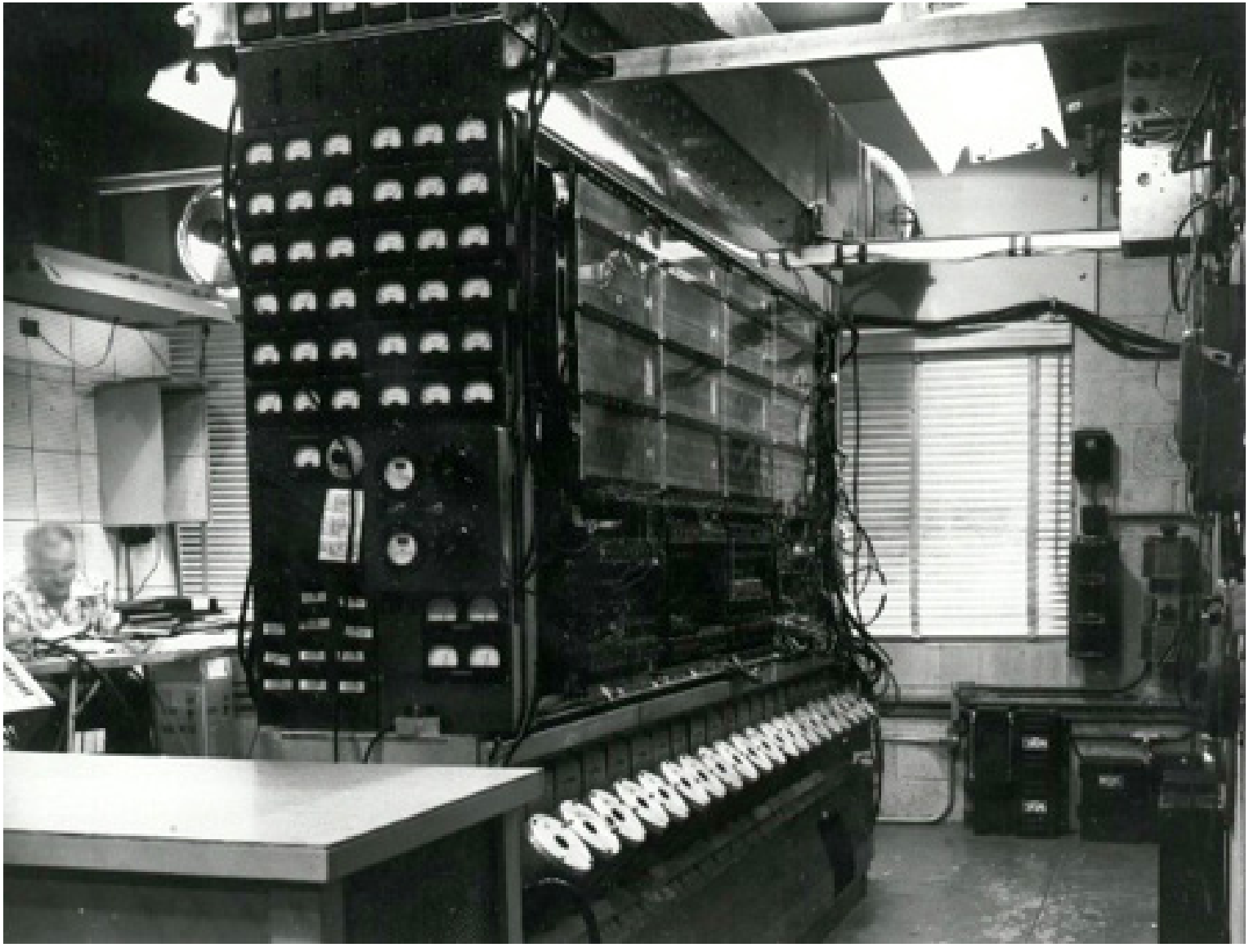
8 marca 2016

Procesor jako maszyna

Altair 8800 (1975)

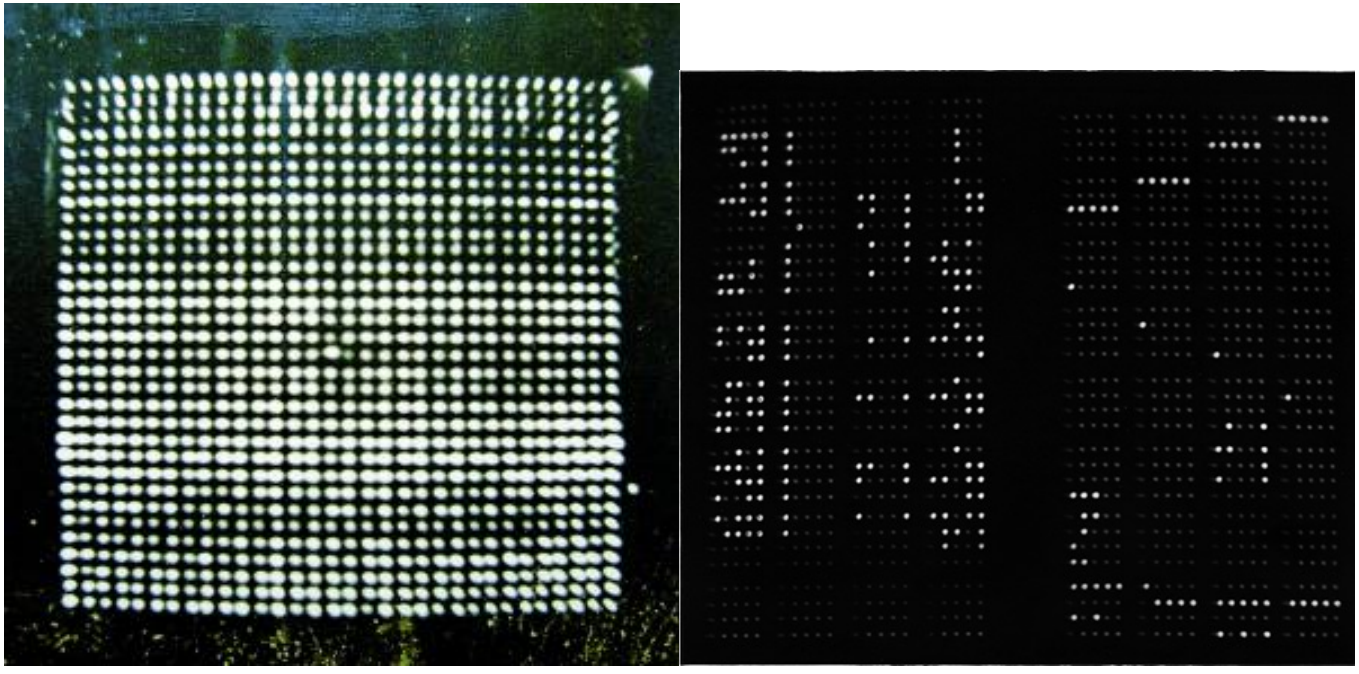
Procesor jako maszyna (2)

Pamięć IAS (ca. 1952)



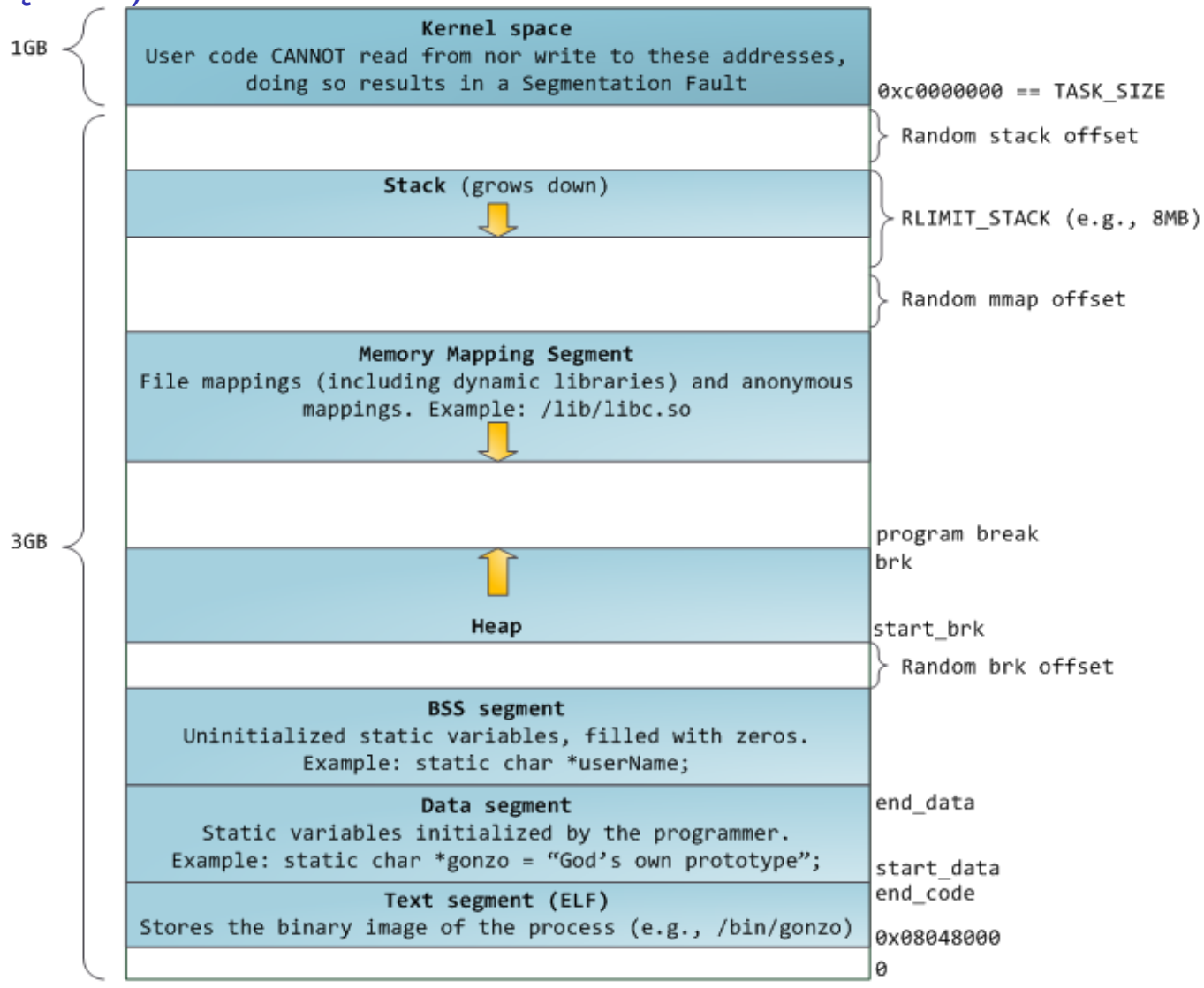
Procesor jako maszyna (2)

Pamięć IAS (ca. 1952)



Mapa pamięci procesu

(poglądowa)



- komenda `pmap`

Program – kod

test.s

```
# vim: syntax=gas

.text

hello: .ascii "Something.\n"
hello_len = . - hello

.global _start
_start:

mov hello, %al
mov hello, %eax

mov $1, %eax # exit
mov $0, %ebx # exit code 0
int $0x80
```

Sprawdzenie (za)wartości pamięci

- Ładujemy program (i dane) do pamięci

```
pepe@lenovo:/tmp$ gdb -q test
Reading symbols from test...(no debugging symbols found)...done.
(gdb) break _start
Breakpoint 1 at 0x400083
(gdb) run
Starting program: /tmp/test
(gdb) disassemble _start
Dump of assembler code for function _start:
 0x400083 <+0>:     8a 04 25 78 00 40 00     mov    0x400078,%al
 0x40008a <+7>:     8b 04 25 78 00 40 00     mov    0x400078,%eax
 0x400091 <+14>:    b8 01 00 00 00   mov    $0x1,%eax
 0x400096 <+19>:    bb 00 00 00 00   mov    $0x0,%ebx
 0x40009b <+24>:    cd 80    int    $0x80
End of assembler dump.
```

- Czego się spodziewamy po wykonaniu rozkazu `mov 0x400078, %al`?

Sprawdzenie (za)wartości pamięci (2)

- komenda `x` (examine)

```
pepe@lenovo:/tmp$ gdb -q test
Reading symbols from test...(no debugging symbols found)...done.
(gdb) break _start
Breakpoint 1 at 0x400083
(gdb) run
Starting program: /tmp/test
(gdb) x /c hello
0x400078 <hello>:       83 'S'
(gdb) x /10c hello
0x400078 <hello>:        83 'S'  111 'o' 109 'm' 101 'e'
                        116 't'  104 'h' 105 'i' 110 'n'
0x400080 <hello+8>:     103 'g'   46 '.'
(gdb) x /10bx hello
0x400078 <hello>:       0x53    0x6f    0x6d    0x65
                        0x74    0x68    0x69    0x6e
0x400080 <hello+8>:     0x67    0x2e
```

Sprawdzenie (za)wartości pamięci (3)

```
pepe@lenovo:/tmp$ gdb -q test
Reading symbols from test...(no debugging symbols found)...done.
(gdb) break _start
Breakpoint 1 at 0x400083
(gdb) run
Starting program: /tmp/test
Breakpoint 1, 0x0000000000400083 in _start ()
(gdb) si
0x000000000040008a in _start ()
(gdb) print /x $al
$1 = 0x53
```

Bajty i słowa

```
pepe@lenovo:/tmp$ gdb -q test
Reading symbols from test...(no debugging symbols found)...done.
(gdb) break _start
Breakpoint 1 at 0x400083
(gdb) run
Starting program: /tmp/test
Breakpoint 1, 0x0000000000400083 in _start ()
(gdb) si
0x000000000040008a in _start ()
(gdb) si
(gdb) x /10bx hello
0x400078 <hello>:       0x53    0x6f    0x6d    0x65
                        0x74    0x68    0x69    0x6e
0x400080 <hello+8>:     0x67    0x2e
0x0000000000400091 in _start ()
(gdb) print /x $eax
$1 = 0x656d6f53
```

- Uporządkowanie little endian
  - Bajty mniej znaczące pod niższymi adresami

Bajty i słowa (2)

```
pepe@lenovo:/tmp$ gdb -q test
Reading symbols from test...(no debugging symbols found)...done.
(gdb) break _start
Breakpoint 1 at 0x400083
(gdb) run
Starting program: /tmp/test
Breakpoint 1, 0x0000000000400083 in _start ()
(gdb) x /12lx hello
0x400078 <hello>:       0x53    0x6f    0x6d    0x65
                        0x74    0x68    0x69    0x6e
0x400080 <hello+8>:     0x67    0x2e    0x0a    0x8a
(gdb) x /3xw hello
0x400078 <hello>:       0x656d6f53      0x6e696874
                        0x8a0a2e67
```

- 0x0a – znak nowej linii
- 0x8a – ??