Übung zu Betriebssysteme

Bildschirm (CGA Textmode)

Wintersemester 2020/21

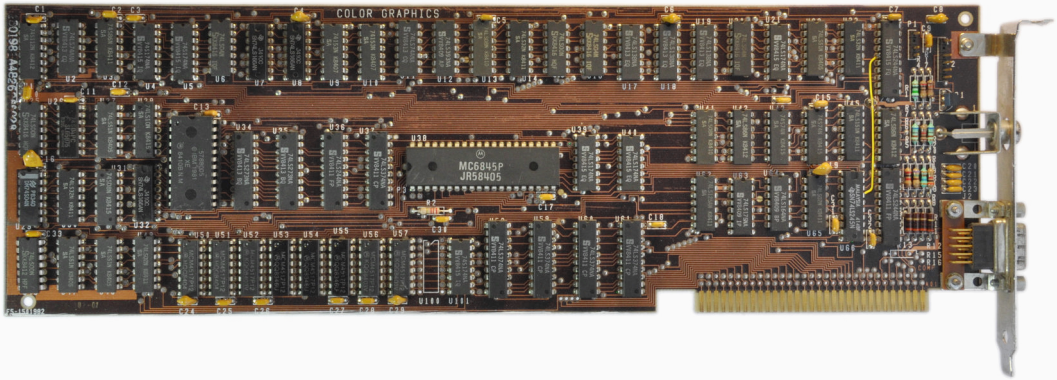
Bernhard Heinloth & Christian Eichler

Lehrstuhl für Informatik 4
Friedrich-Alexander-Universität Erlangen-Nürnberg

Lehrstuhl für Verteilte Systeme
und Betriebssysteme

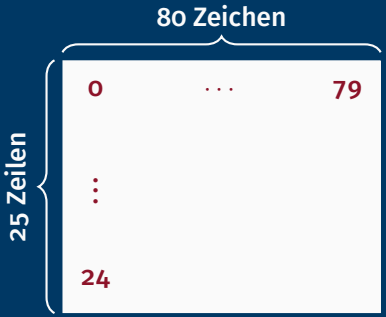
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT



Quelle: Wikipedia

Der CGA Bildschirm ...



… zwei Bytes pro Zeichen …

1. Byte: **ASCII Zeichen**

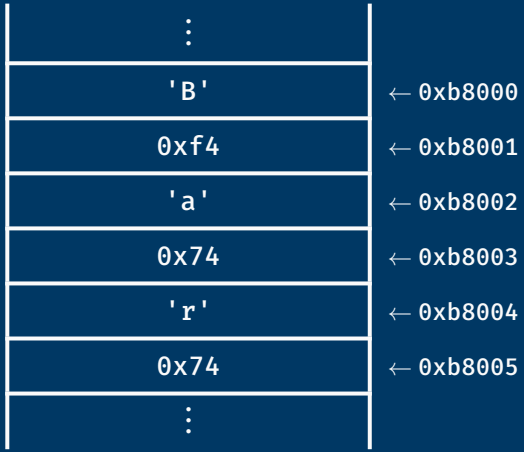
... zwei Bytes pro Zeichen ...

1. Byte: **ASCII Zeichen**

2. Byte: **Darstellungsattribut**

0	schwarz	4	rot	8	dunkelgrau	12	hellrot
1	blau	5	magenta	9	hellblau	13	hellmagenta
2	grün	6	braun	10	hellgrün	14	gelb
3	cyan	7	hellgrau	11	hellcyan	15	weiß

...eingeblendet im Arbeitsspeicher ab `0xb8000`




Bar

Ansprechen von einzelnen Bits

- Bitmasken und logischen Bitoperationen

`&`	Konjunktion (bitweises UND)	`3 & 5 = 1`
`|`	Disjunktion (bitweises ODER)	`3 | 5 = 7`
`^`	exklusives ODER	`3 ^ 5 = 6`
`~`	Einerkomplement (bitweise Negation)	`~ 5 = 250`
`<<`	verschieben nach links	`12 << 2 = 48`
`>>`	verschieben nach rechts	`12 >> 2 = 3`

`x |= (1 << n)`	Setze das `n`te Bit in `x`
`x &= ~(1 << n)`	Lösche das `n`te Bit in `x`

Ansprechen von einzelnen Bits

- Bitmasken und logischen Bitoperationen
- Bitfelder in C/C++

```
struct cga_attrib {
    unsigned char fg : 4,
                  bg : 3,
                  bl : 1;
};

struct cga_attrib a;
a.fg = 4;  // rot
a.bg = 7;  // hellgrau
a.bl = 1;  // blinken
```

Layout mit GCC auf x86:

Einschub: Füllen von Strukturen

```
struct Test {
    char foo[3]; // 3 Byte
    int bar;     // 4 Byte
    bool baz;    // 1 Byte
} __attribute__((packed));

static_assert(sizeof(Test) == 8, "Test kaputt");
```

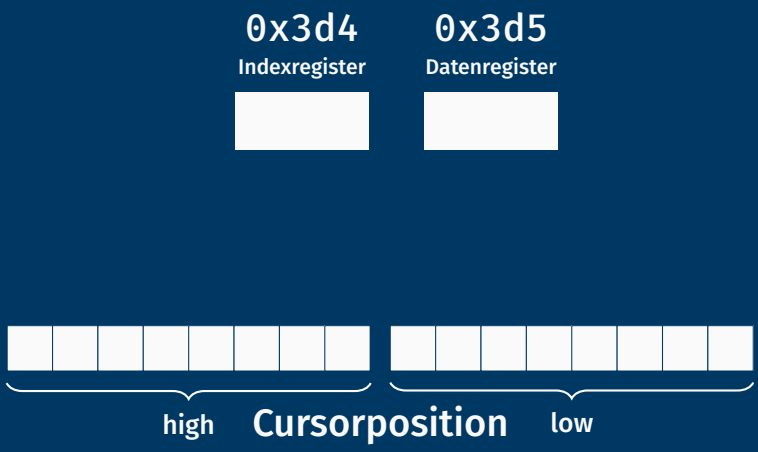
Schreibmarke (Cursor)

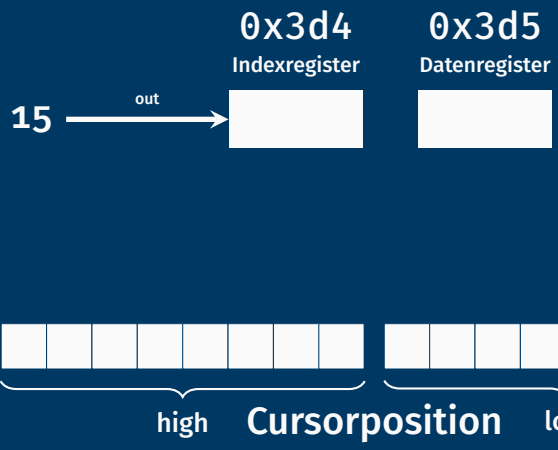
Entweder Cursorposition in Software merken...

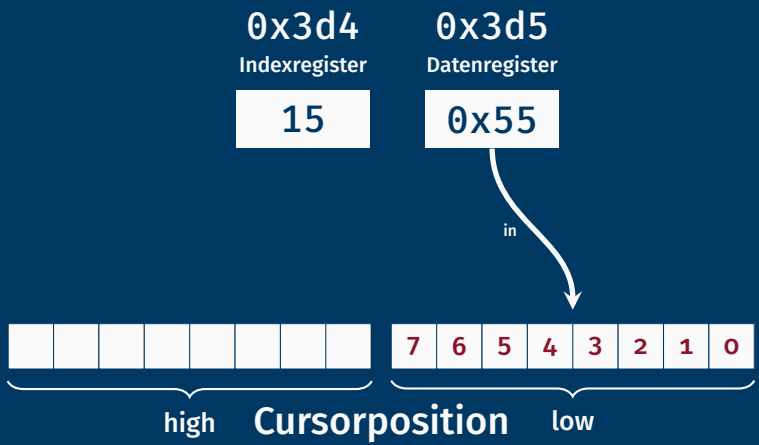
… oder Cursorposition aus Hardware auslesen

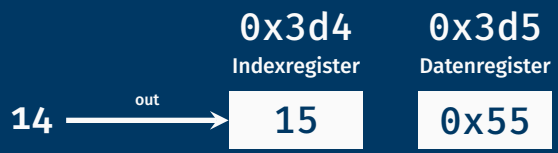
- CGA hat 18 Steuerregister mit je 8 Bit
- Position in Register 14 (high) und 15 (low)
- Nur indirekter Zugriff über Index- (`0x3d4`) und Datenregister (`0x3d5`)

0	Horizontal Total	w
1	Horizontal Displayed	w
2	H. Sync Position	w
3	H. Sync Width	w
4	Vertical Total	w
5	V. Total Adjust	w
6	Vertical Displayed	w
7	V. Sync Position	w
8	Interlace Mode	w
9	Max Scan Line Address	w
10	Cursor Start	w
11	Cursor End	w
12	Start Address (high)	w
13	Start Address (low)	w
14	Cursor (high)	rw
15	Cursor (low)	rw
16	Light Pen (high)	r
17	Light Pen (low)	r

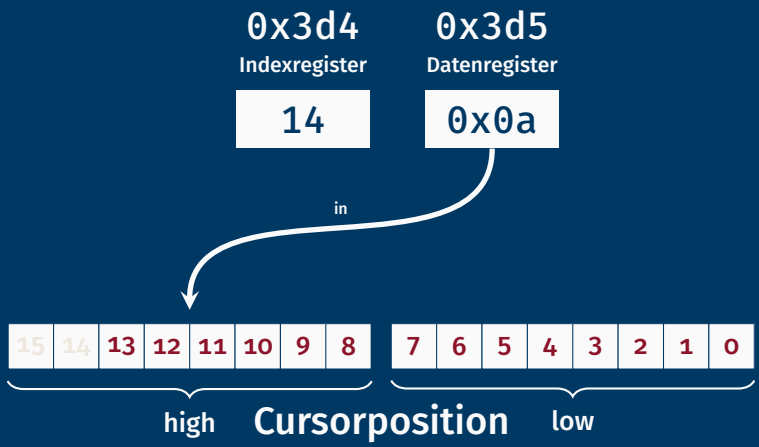

0x3d4
Indexregister
0x3d5
Datenregister
high
Cursorposition
low


0x3d4
Indexregister
0x3d5
Datenregister
15
out
high
Cursorposition
low

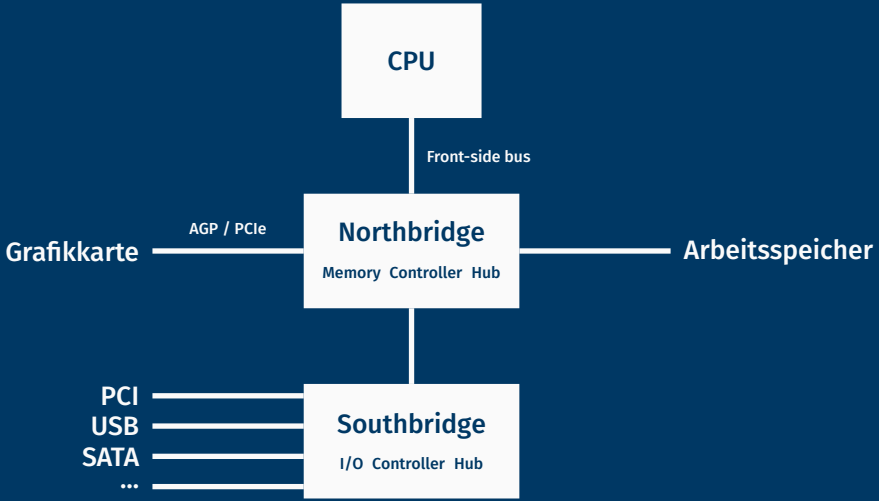

0x3d4
Indexregister
0x3d5
Datenregister
15
0x55
in
7 6 5 4 3 2 1 0
high
Cursorposition
low

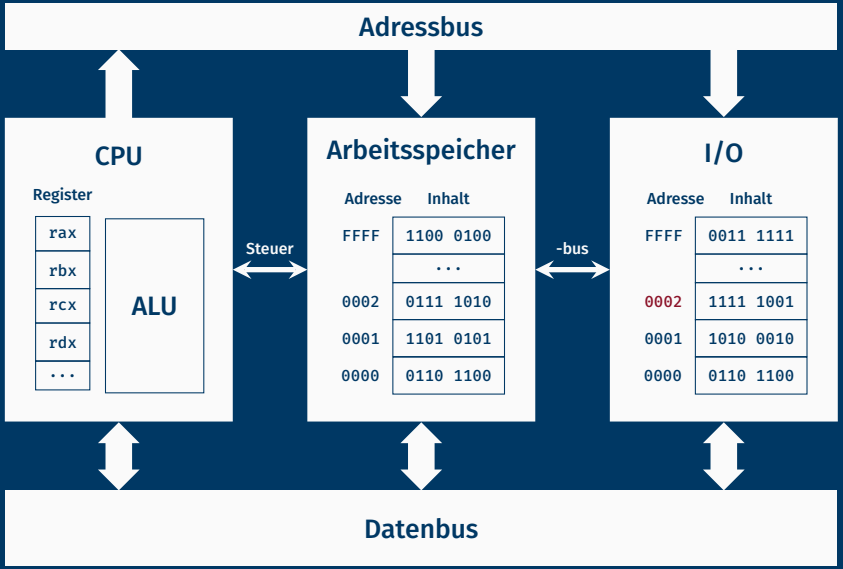

0x3d4
Indexregister
0x3d5
Datenregister
14
out
15
0x55


7 6 5 4 3 2 1 0
high
Cursorposition
low


0x3d4
Indexregister
0x3d5
Datenregister
14
0x0a
in
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
high
Cursorposition
low

Kleiner Exkurs:
Zugriff auf Arbeitsspeicher vs. I/O


CPU
Front-side bus
AGP / PCIe
Grafikkarte
Northbridge
Memory Controller Hub
Arbeitsspeicher
PCI
USB
SATA
...
Southbridge
I/O Controller Hub


Adressbus
CPU
Register
rax
rbx
rcx
rdx
...
ALU
Steuer
Arbeitsspeicher
Adresse
Inhalt
FFFF
1100 0100
...
0002
0111 1010
0001
1101 0101
0000
0110 1100
-bus
I/O
Adresse
Inhalt
FFFF
0011 1111
...
0002
1111 1001
0001
1010 0010
0000
0110 1100
Datenbus