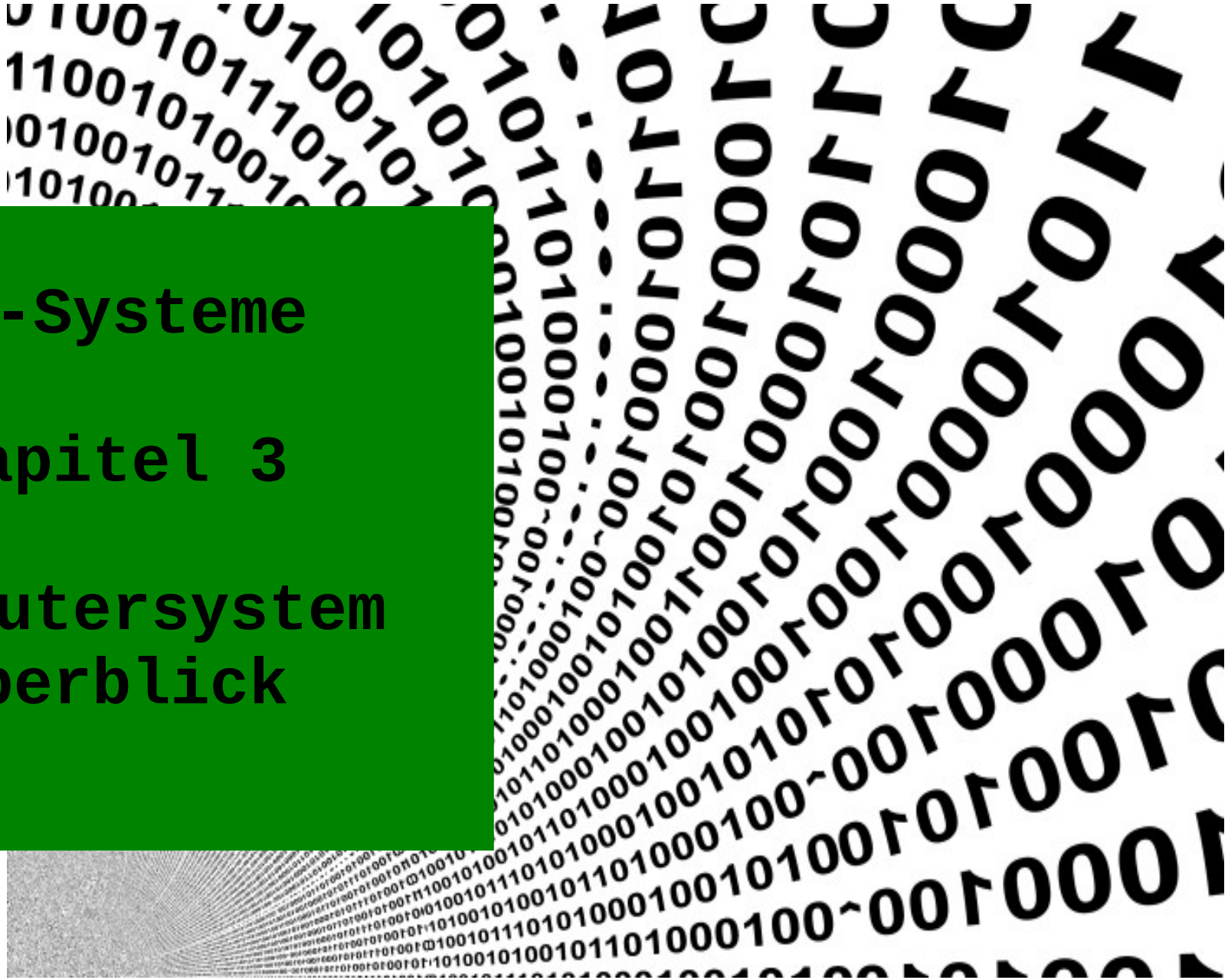


DS-Systeme

Kapitel 3

Computersystem Überblick

Inhaltsverzeichnis

Thema	Seite
Metrische Einheiten	3
Datenspeicher - Einheiten	7
Aufbau eines Computers	9
Fetch–Decode–Execution - Zyklus	12
Was passiert beim "Hello World!" - Programm?	15
Prozessor	18
Hauptbestandteile der CPU	19
(CPU-)Register	21
General Purpose Register	22
Aufgabe zu General Purpose Registern	24
Flag-Register	26
xmm-Register	27
xmm-, ymm und zmm-Register	28

Metrische Einheiten

Exp.	ausgeschrieben	Präfix
10^{-3}	0.001	Milli
10^{-6}	0.000 001	Mikro
10^{-9}	0.000 000 001	Nano
10^{-12}	0.000 000 000 001	Pico
10^{-15}	0.000 000 000 000 001	Femto
10^{-18}	0.000 000 000 000 000 001	Atto
10^{-21}	0.000 000 000 000 000 000 001	Zepto
10^{-24}	0.000 000 000 000 000 000 000 001	Yokto

Aufgaben:

1. Wieviele Nanometer sind ein Millimeter? 1 mm = ? nm
2. Wieviele Mikrometer sind ein Femtometer? 1 fm = ? µm

Geben Sie die Antworten sowohl als Potenz als auch als Dezimalzahl an.

Metrische Einheiten

Exp.	ausgeschrieben	Präfix
10^{3}	1.000	Kilo
10^{6}	1.000 000	Mega
10^{9}	1.000 000 000	Giga
10^{12}	1.000 000 000 000	Tera
10^{15}	1.000 000 000 000 000	Peta
10^{18}	1.000 000 000 000 000 000	Exa
10^{21}	1.000 000 000 000 000 000 000	Zetta
10^{24}	1.000 000 000 000 000 000 000 000	Yotta

Aufgabe:

SSD-Festplatten werden meistens in den Grössen 250GB, 500GB, 1TB, 2TB und 4TB angeboten. Wieviel Kilobyte sind ein Terabyte?

1 TB = ? KB

Metrische Einheiten

Da **Kilo** für 1000 steht, **Mega** für 1.000.000 u.s.w., sind die Angaben der vorherigen Seite eigentlich irreführend. Da Byte-Angaben immer in 2-er-Potenzen angegeben werden, müssten sie eigentlich IEC-konform folgendermaßen angegeben werden:

										Abkürzungen:
1	Kilobyte	=	10^3	Byte,	1	Kibibyte	=	2^{10}	Byte	KB / KiB
1	Megabyte	=	10^6	Byte,	1	Mebibyte	=	2^{20}	Byte	MB / MiB
1	Gigabyte	=	10^9	Byte,	1	Gibibyte	=	2^{30}	Byte	GB / GiB
1	Terabyte	=	10^{12}	Byte,	1	Tebibyte	=	2^{40}	Byte	TB / TiB
1	Petabyte	=	10^{15}	Byte,	1	Pebibyte	=	2^{50}	Byte	PB / PiB
1	Exabyte	=	10^{18}	Byte,	1	Exbibyte	=	2^{60}	Byte	EB / EiB
1	Zetabyte	=	10^{21}	Byte,	1	Zebibyte	=	2^{70}	Byte	ZB / ZiB
1	Yottabyte	=	10^{24}	Byte,	1	Yobibyte	=	2^{80}	Byte	YB / Yib

IEC-konform

Merkregel:

Die ersten beiden Buchstaben aus der ursprünglichen Bezeichnung bleiben erhalten, der Rest wird durch "**bibyte**" ersetzt. Bei der Abkürzung wird jeweils ein kleines "**i**" eingeschoben, z.B.: **MB → MiB**.

Metrische Einheiten

Entsprechendes gilt für die Einheiten bezüglich einzelner Bits.

Multiple-bit units V·T·E

Decimal			Binary				
Value	Metric		Value	IEC		Legacy	
1000	kbit	kilobit	1024	Kibit	**kibibit**	Kbit	Kb kilobit
1000^2	Mbit	megabit	1024^2	Mibit	mebibit	Mbit	Mb megabit
1000^3	Gbit	gigabit	1024^3	Gibit	gibibit	Gbit	Gb gigabit
1000^4	Tbit	terabit	1024^4	Tibit	tebibit	Tbit	Tb terabit
1000^5	Pbit	petabit	1024^5	Pibit	pebibit		—
1000^6	Ebit	exabit	1024^6	Eibit	exbibit		—
1000^7	Zbit	zettabit	1024^7	Zibit	zebibit		—
1000^8	Ybit	yottabit	1024^8	Yibit	yobibit		—

Orders of magnitude of data

Quelle: Wikipedia

Aufgabe:

Wieviel Megabyte sind ein Tebibit?

1 Tibit = ? MB

Datenspeicher-Einheiten

Die *x86-64 Architektur unterstützt eine bestimmte Menge von Datengrößen – alle auf der Basis von 2^n.

Speichereinheit	Größe in Bits	Größe in Bytes
Byte	8	1
Word	16	2
Double word	32	4
Quadword	64	8
Double quadword	128	16

Diese Größen korrellieren direkt mit den Größen von Datentyp in höheren Programmiersprachen wie C, C++, Java, u.s.w.

* In der Informatik wird die auf dem x86-Befehlssatz (ISA) basierende 64-Bit-Architektur als x64, alternativ auch als x86-64 und AMD64 bezeichnet. Die Befehlssatzerweiterung ergänzt die Intel Architecture 32-Bit, kurz IA-32, um einen 64-Bit-Betriebsmodus.

Datenspeicher-Einheiten

In C und C++ zum Beispiel entspricht dies den folgenden Größen:

C/C++ Deklaration	Speichereinheit	Bits	Bytes
char	Byte	8	1
short	Word	16	2
int	Double word	32	4
unsigned int	Double word	32	4
long	*Double word	*32	*4
long long	Quadword	64	8
char *	Quadword	64	8
int *	Quadword	64	8
float	Double word	32	4
double	Quadword	64	8

*Für den C- bzw. C++-Datentyp **long** ist hier ein **Double word** eingetragen (Windows).
Der **Intel**-Datentyp **long** (Verwendung in GNU-Assembler) ist dort als **Quadword** vorgesehen, weil er unter Linux 8 Byte verbraucht.

Aufbau eines Computers

Hauptunterscheidung
- Hardware (DV-Anlage, Geräte)
- Software (darauf ablaufende Programme)

Software

Betriebssytem
- Windows, Linux, MacOS, Android, iOS, u.s.w.

Anwendungssoftware
- Textverarbeitung, Spiele- , Büro-Software, Mathematische Software, IDE's, u.s.w.

Hardware

Feste Komponenten, die das Gerät zum Laufen bringen.

Peripherie (äußerlich sichtbar)

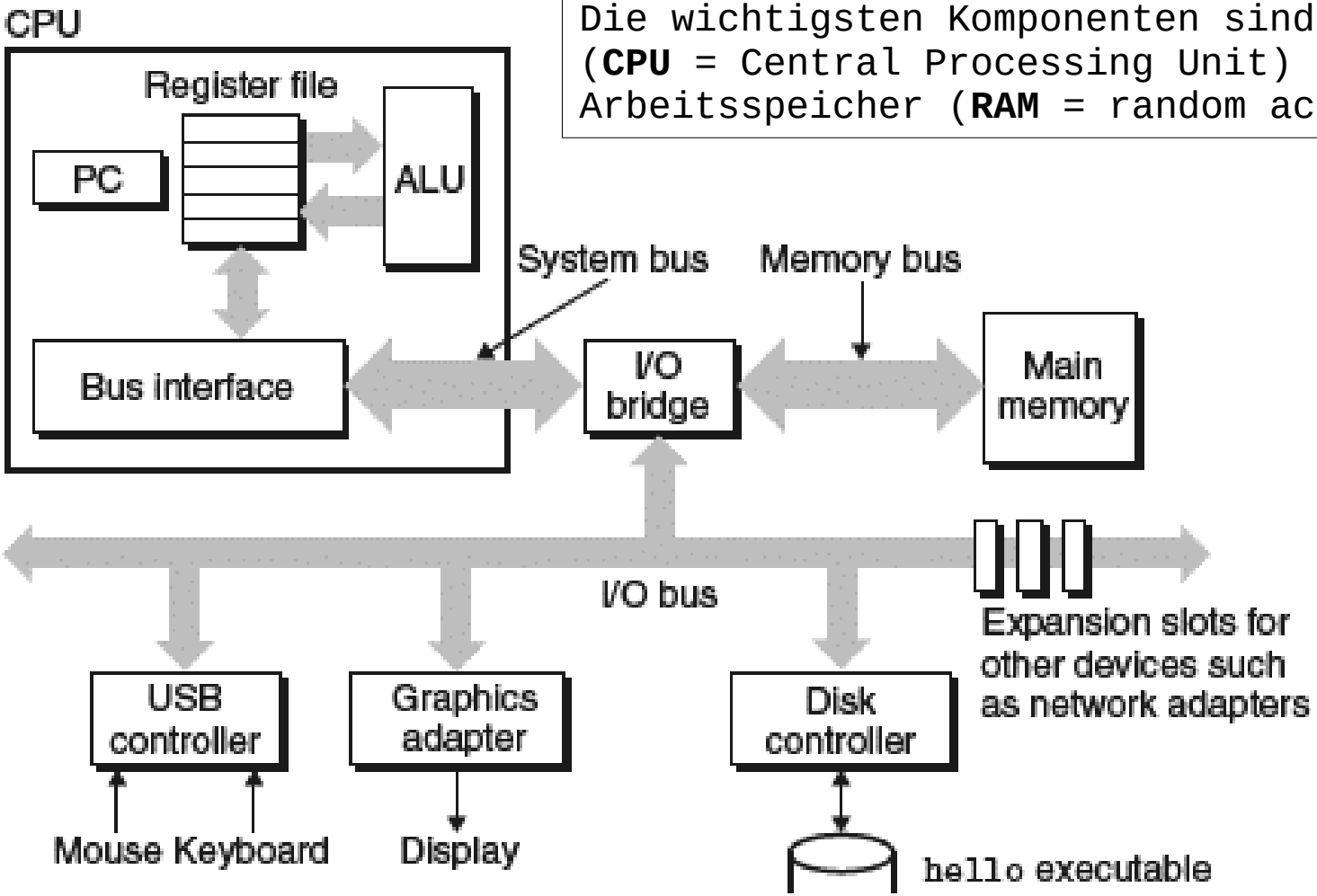
- Ein-/Ausgabegeräte
  Bildschirm, Tastatur, Maus, Joystick, externe Festplatten, u.s.w.

Hardwarekomponenten (innen)

- Prozessor
- Primärspeicher
- Sekundärspeicher
- Mainboard
- Graphikkarte
- Netzwerkkarte
- Bussystem

Aufbau eines Computers

Um zu verstehen, was beim Ausführen abläuft, muss man die Hardware-Organisation eines Computer-Systems verstehen.
Die wichtigsten Komponenten sind dabei der Prozessor (**CPU** = Central Processing Unit) und der Haupt- bzw. Arbeitsspeicher (**RAM** = random access memory).



Aufbau eines Computers

Die **CPU** macht eigentlich nichts anders als:

- Assemblerbefehle bzw. Maschinenbefehle (instructions) aus dem Hauptspeicher zu laden, auf die der Program Counter (PC) aktuell zeigt. (Fetch-Phase)

- Befehle dekodieren (Dekodier-Phase)

- Befehle ausführen (Execute-Phase)

- PC (Program Counter) erhöhen

- nächsten Befehl laden
  u.s.w.

Die Befehle geben an, wie die Hauptkomponenten der CPU (Register File, Rechenwerk, Steuerwerk) konfiguriert und verwendet werden sollen.

Beispiele: load, store, operate (z.B. add), jump; u.s.w.

Fetch – Decode – Execute - Zyklus

Grundsätzlich redet man bei den **CPU-Befehlsabarbeitungsphasen** vom

Fetch – Decode – Execute - Cycle

Dabei werden manchmal noch zusätzliche Phasen angegeben - wie hier am Beispiel einer klassischen **RISC**-Pipeline (**R**educed **I**nstruction **S**et **C**omputer).

- Phase **Instruction Fetch (IF)**
  Instruktion (Befehl) aus dem (Programm-)Speicher holen

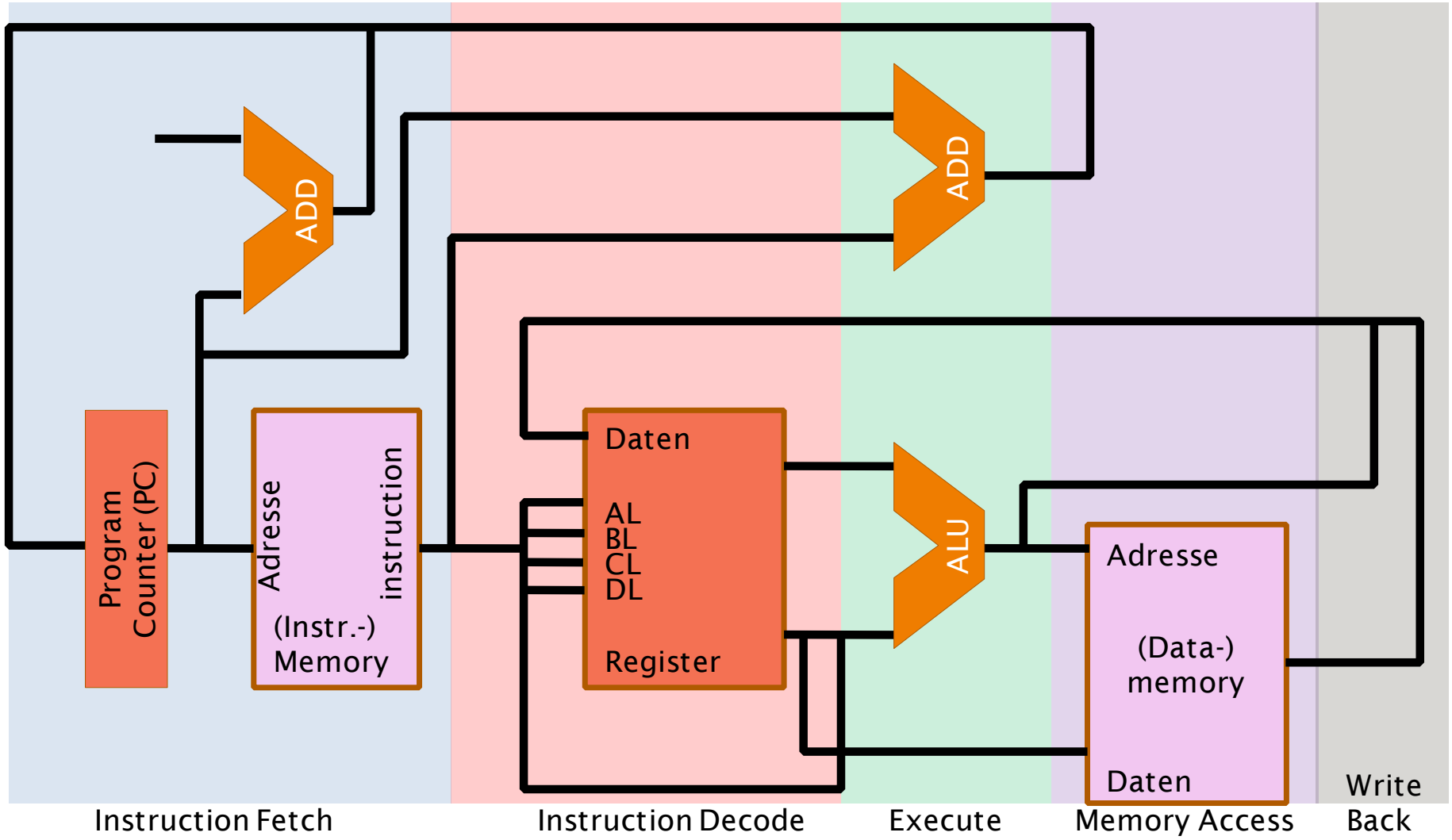
- Phase **Instruction Decode (ID)**
  - Instruktion entschlüsseln
  - Operanden bereitstellen
  - (Steuerleitungen setzen)

- Phase **Execute (EX)**
  - Instruction ausführen (ALU)

- Phase **Memory Access (MEM)**
  Daten aus (Daten-)speicher holen / zum Datenspeicher schaffen

- Phase **Write Back (WB)**
  Register mit Ergebnis / speicherinhalt befüllen

Fetch – Decode – Execute - Zyklus



Die rosa eingefärbten Rechtecke liegen außerhalb der CPU (Instr.-Memory, Data-Memory).

Fetch – Decode – Execute - Zyklus

Instruction Fetch:

Program Counter (PC) legt Adresse an (Instruction-Memory) -> Laden des Befehls

- PC wird mit linkem Addierer (ADD) um 1 erhöht

Instruction Decode:

die Operanden des Befehls werden geladen (z.B. reg, imm)

Execute:

Operanden werden mit der ALU gemäß des Befehls verarbeitet

Memory Access:

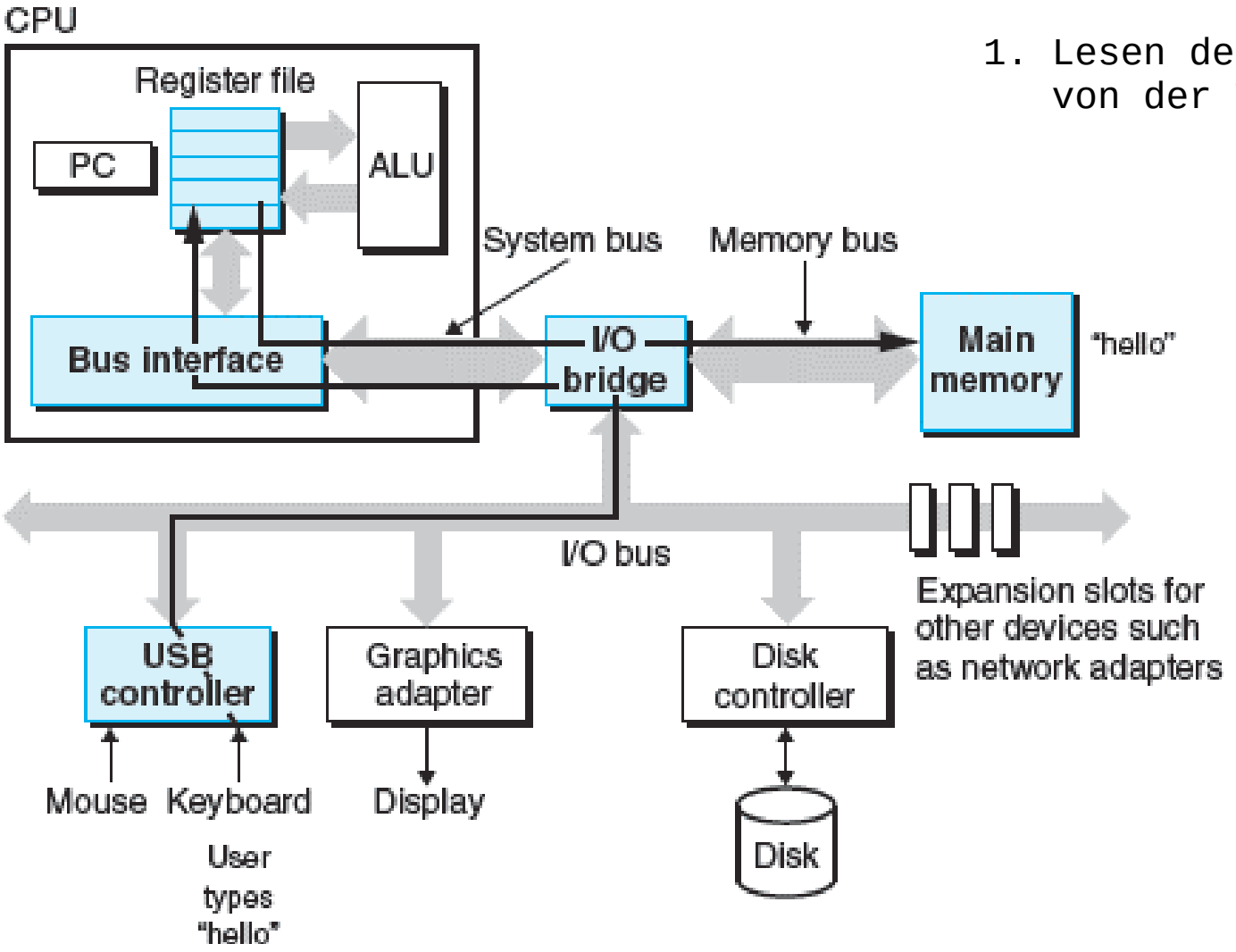
ggf. wird das Ergebnis in den Arbeitsspeicher geschrieben (store) oder die gewünschten Daten in Register geladen (load)

Write Back:

ggf. wird Ergebnis wird zurück ins Register geschrieben

(der oberer Pfad ist für jumps)

Was passiert beim Hello World-Programm?



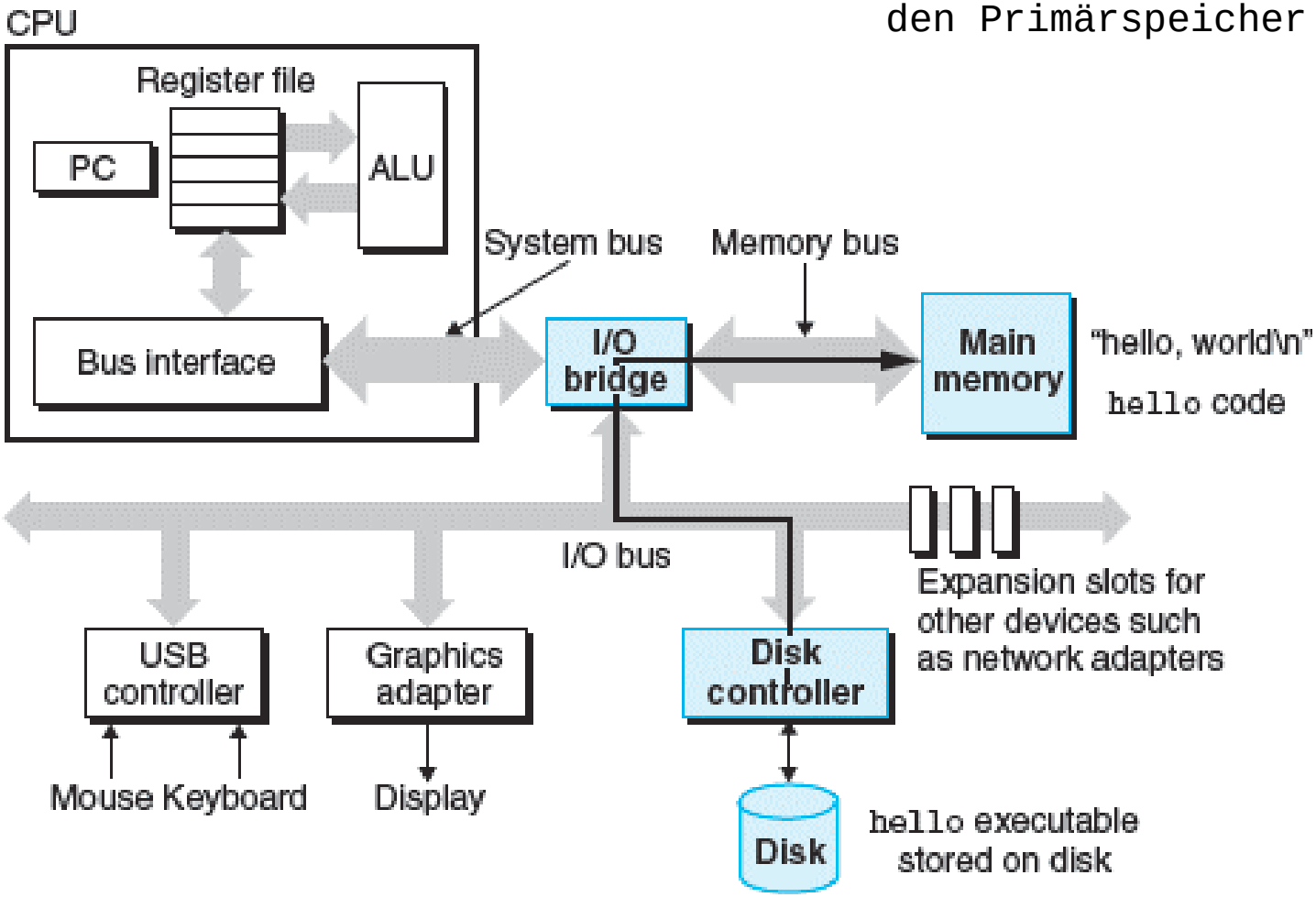
1. Lesen des Befehls **./hello** von der Tastatur.

Hellblau markierte Elemente sind gerade an der Aktion beteiligt.

Die **schwarzen** Linien bzw. Pfeile zeigen die aktuell beteiligten Bussysteme an.

Was passiert beim Hello World-Programm?

2. Laden der ausführbaren Datei **hello** vom Sekundärspeicher in den Primärspeicher.

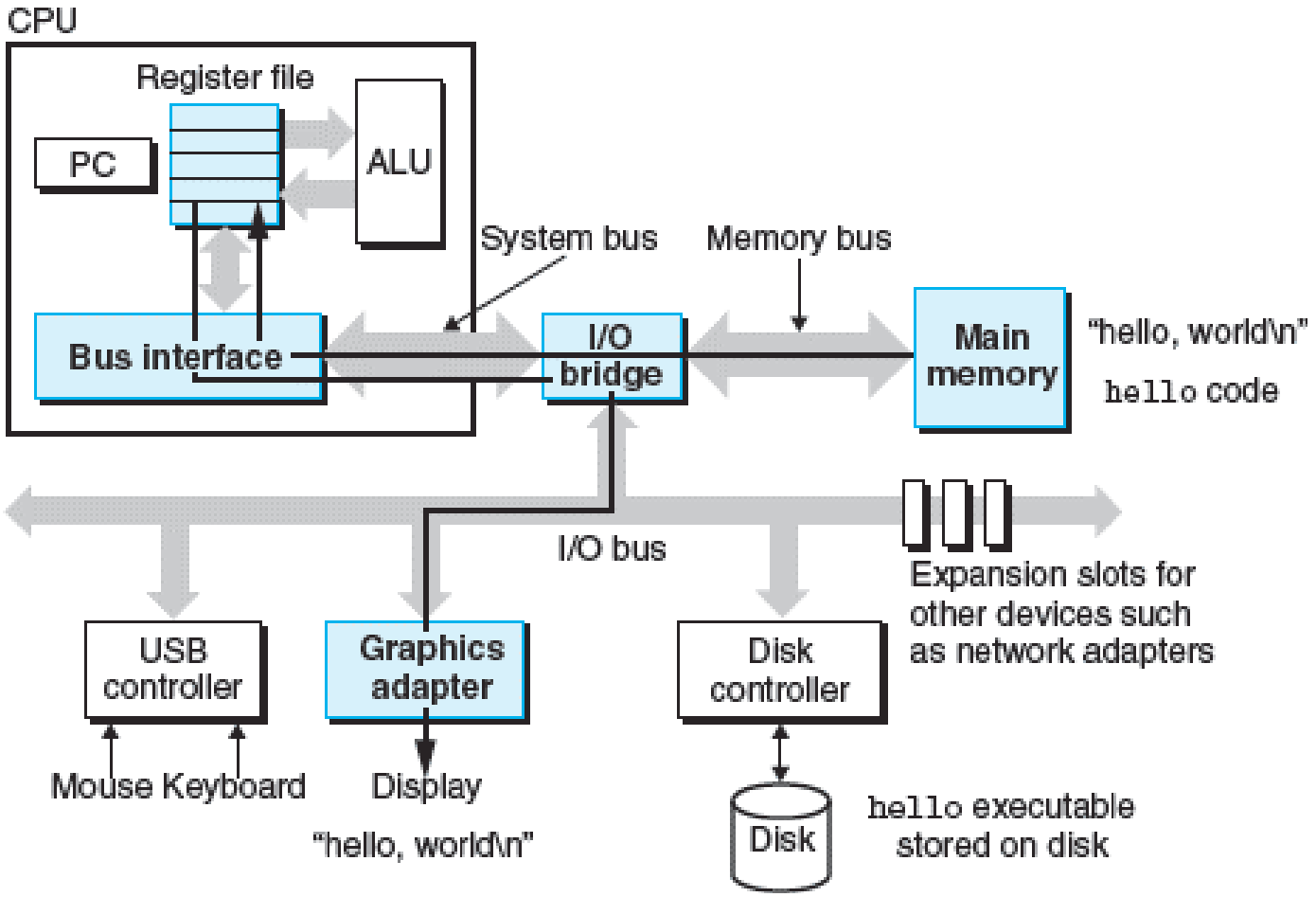


Hellblau markierte Elemente sind gerade an der Aktion beteiligt.

Die **schwarzen** Linien bzw. Pfeile zeigen die aktuell beteiligten Bussysteme an.

Was passiert beim Hello World-Programm?

3. Schreiben der Ausgabe **"Hello World!"** vom Hauptspeicher auf das Display.



Hellblau markierte Elemente sind gerade an der Aktion beteiligt.

Die **schwarzen** Linien bzw. Pfeile zeigen die aktuell beteiligten Bussysteme an.

Prozessor

CPU (Central Processing Unit, Hauptprozessor, Zentraleinheit)

- befindet sich auf speziellen Sockel auf dem Mainboard
- ist mit zusätzlichen Lüfter für die Kühlung bestückt, da wegen der hohen Leistung Hitze entstehen kann
- führende Hersteller: AMD und Intel

- führt entsprechend dem Prinzip des von Neumann-Rechners sämtliche Berechnungen durch, die für den Betrieb des Computers benötigt werden

Intel-Prozessor

AMD-Prozessor

Hauptbestandteile der CPU:

- Steuerwerk
- Rechenwerk
- Register
- Cache-Speicher
- Taktgeber
- Bussystem

Hauptbestandteile der CPU

Steuerwerk (Leitwerk / Verwaltungseinheit)

- Steuerung des Ablaufs der Befehlszyklen in der CPU
  - Steuerung des Programmablaufs
- Steuerung der Befehlsausführung
  - Befehle holen und decodieren
  - Operanden holen / speichern
- Koordination und Steuerung der einzelnen Teile der CPU über den Steuerbus

Rechenwerk

- arithmetische Operationen (Addition, Subtraktion, ...)
- logische Operationen (UND, ODER, NICHT, ...)
- Verschiebe-Operationen
- u. U. Bitmanipulation
- Vergleichs- und Bit-Test-Operationen

Aufgabe Rechenwerk:

Führen Sie folgende Potenzrechnung schrittweise auf eine Addition zurück:

$2^3 + 3^2$

Hauptbestandteile der CPU

Rechenwerk **einfacher** CPUs:

- Rechenwerk besteht aus einer einzigen Einheit für die Durchführung aller Operationen.
- alle arithmetischen Operationen werden auf die Addition zurückgeführt

Rechenwerk **komplexerer** CPUs:

- haben ein Rechenwerk, das aus mehreren Unterwerken für die einzelnen Operationen besteht (schnellere, parallele Abarbeitung möglich) (z. B. Addition, Subtraktion, Multiplikation, Division, Gleitkomma-Addition, Normalisierung, Increment, log. Verknüpfung, Verschiebung)
- es kann spezielle Prozessoren für bestimmte Operationen geben (z.B. Gleitkommaprozessoren)

Die **ALU** (Arithmetic and Logic Unit) bildet den Kern des Rechenwerks. Sie kann einfache arithmetische unbd logische Operationen durchführen.

(CPU-)Register

Ein CPU-Register oder einfach **Register** ist ein temporärer Speicher bzw. Arbeitsort innerhalb der CPU, der dort getrennt vom Arbeitsspeicher verbaut ist. Berechnungen werden typischerweise von CPU-Registern durchgeführt.

Vorteile:

- Sehr schnelle prozessorientierte Speicher
- Stellen extrem schnelle Verbindungen zu anderen Prozessorteilen bereit.
- Nur Daten, die in Registern stehen, können direkt abgearbeitet werden.

General Purpose Register (Universalregister)

Es gibt sechzehn 64-Bit-Universalregister (GPRs). Sie sind in die folgende Tabelle beschrieben . Auf ein GPR-Register kann mit allen 64-Bits oder einer Teilmenge davon zugegriffen werden. Um auf die Teilmengen zugreifen zu können, werden andere Namen verwendet (s. Tabelle)

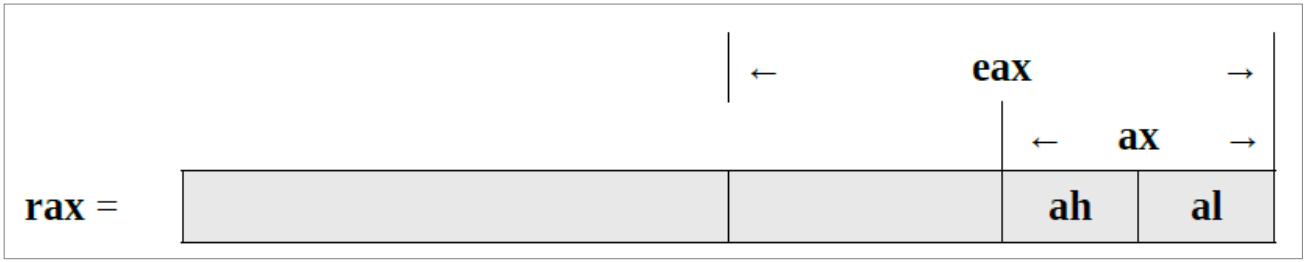
Des weiteren werden einige dieser Register für ganz bestimmte Zwecke eingesetzt.

General Purpose Register (Universalregister)

64-bit register	Lowest 32-bits	Lowest 16-bits	Lowest 8-bits
rax	eax	ax	al
rbx	ebx	bx	bl
rcx	ecx	cx	cl
rdx	edx	dx	dl
rsi	esi	si	sil
rdi	edi	di	dil
rbp	ebp	bp	bpl
rsp	esp	sp	spl
r8	r8d	r8w	r8b
r9	r9d	r9w	r9b
r10	r10d	r10w	r10b
r11	r11d	r11w	r11b
r12	r12d	r12w	r12b
r13	r13d	r13w	r13b
r14	r14d	r14w	r14b
r15	r15d	r15w	r15b

General Purpose Register (Universalregister)

Bei Verwendung von Datenelementgrößen von weniger als 64 Bit (d. h. 32, 16 bzw. 8 Bit) kann auf den niedrigwertigeren Teil des Registers zugegriffen werden, indem ein anderer Registername verwendet wird.



In den ersten vier Registern **rax**, **rbx**, **rcx** und **rdx** kann auch auf die Bits 8–15 mit den Registernamen **ah**, **bh**, **ch** und **dh** zugegriffen werden. Mit Ausnahme von **ah**, werden sie für *Legacy-Unterstützung bereitgestellt.

*Legacy-Unterstützung:

Ersatz-BIOS für Rechner, bei denen das neue UEFI (Unified Extensible Firmware Interface) – Nachfolger vom BIOS – nicht richtig funktioniert.

Aufgabe zu General Purpose Registern

Lösen Sie die folgende Aufgabe unter Verwendung eines Taschenrechners – vorzugsweise Windows-Rechner (Einstellung: Programmierer).

Stellen Sie fest, welcher Inhalt nach diesen drei Operationen im **rax** steht. Geben Sie das Ergebnis in hexadezimaler Form und in der korrekten Breite an.

Gehen Sie dabei davon aus, dass bei jeder Beschreibung eines Registers, der vorherige Inhalt nur an den angegebenen Stellen überschrieben wird.

rax = $50.000.000.000_{10}$

eax = $50.000.000_{10}$

ax = 50.000_{10}

al = 50_{10}

ah = 5_{10}

Stack Pointer Register

Eines der CPU-Register, **rsp**, wird verwendet, um auf den obersten Inhalt des Stacks zu zeigen. Das rsp-Register sollte ausschließlich dafür und nicht für Daten oder andere Zwecke verwendet werden.

Base Pointer Register (Bottom Pointer Register)

Ein anderes CPU-Register, **rbp**, wird für Funktionen im Stack während Funktionsaufrufen als Basiszeiger verwendet. Das rbp-Register sollte ebenfalls nicht für Daten oder andere Zwecke verwendet werden.

Instruction Pointer Register

Zusätzlich zu den GPRs gibt es noch das **rip**-Register, das auch von der CPU verwendet wird. Es zeigt auf den nächsten auszuführenden Befehl.
Dies kann man auch bei einem Debugger sehen – dort wird auch immer auf den nächsten Befehl gezeigt.

Flag Register

Das Flag-Register (RFlags – 64 Bit, EFlags – 32 Bit), wird für Status- und CPU-Steuerinformationen verwendet. Es wird von der CPU nach jeder Anweisung aktualisiert und ist nicht direkt von Programmen ansprechbar. Dieses Register speichert Statusinformationen über den gerade ausgeführten Befehl. Von den 64 Bits im RFlag-Register sind viele für die zukünftige Verwendung reserviert.

Flag Register

Eine Auswahl sehen Sie hier:

Bit	Symbol	Name	Verwendung
0	CF	Carry	Zeigt an, ob es bei der vorhergehenden Operation einen Übertrag gegeben hat.
2	PF	Parity	Zeigt an, ob die Anzahl der 1en im letzten Byte gerade (1) oder ungerade (0) ist.
4	AF	Adjust	Zeigt an, ob es einen Übertrag nach einem halben Byte auf das nächste halbe Byte gegeben hat.
6	ZF	Zero	Zeigt an, ob das Ergebnis der vorhergehenden Operation 0 ist.
7	SF	Sign	Stellt fest, ob das MSB (Most Significant Bit) 1 oder 0 ist (1 = negative Zahl)
10	DF	Direction	Zeigt die Richtung der Zeichenfolgenverarbeitung an.
11	OF	Overflow	Überprüft, ob bei der vorhergehenden Operation der Zahlenbereich eines Datentyps überschritten wurde.

XMM-Register

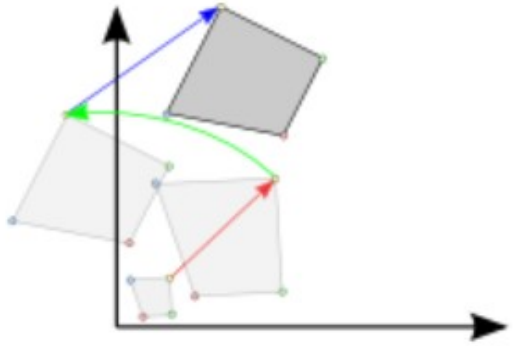
XMM0 bis **XMM15** sind 128-Bit-Register zur Unterstützung von

- 32- und 64-Bit-Gleitkommazahlen-Operationen

- für **SIMD**-Befehle (**Single Instruction Multiple Data**)
  SIMD-Anweisungen ermöglichen die gleichzeitige Anwendung einer einzelnen Anweisung auf mehrere Daten.

  Effektiv eingesetzt, kann dies zu einer deutlichen Leistungssteigerung führen.

  Typische Anwendungen:
  - Grafikverarbeitung
  - digitale Signalverarbeitung.

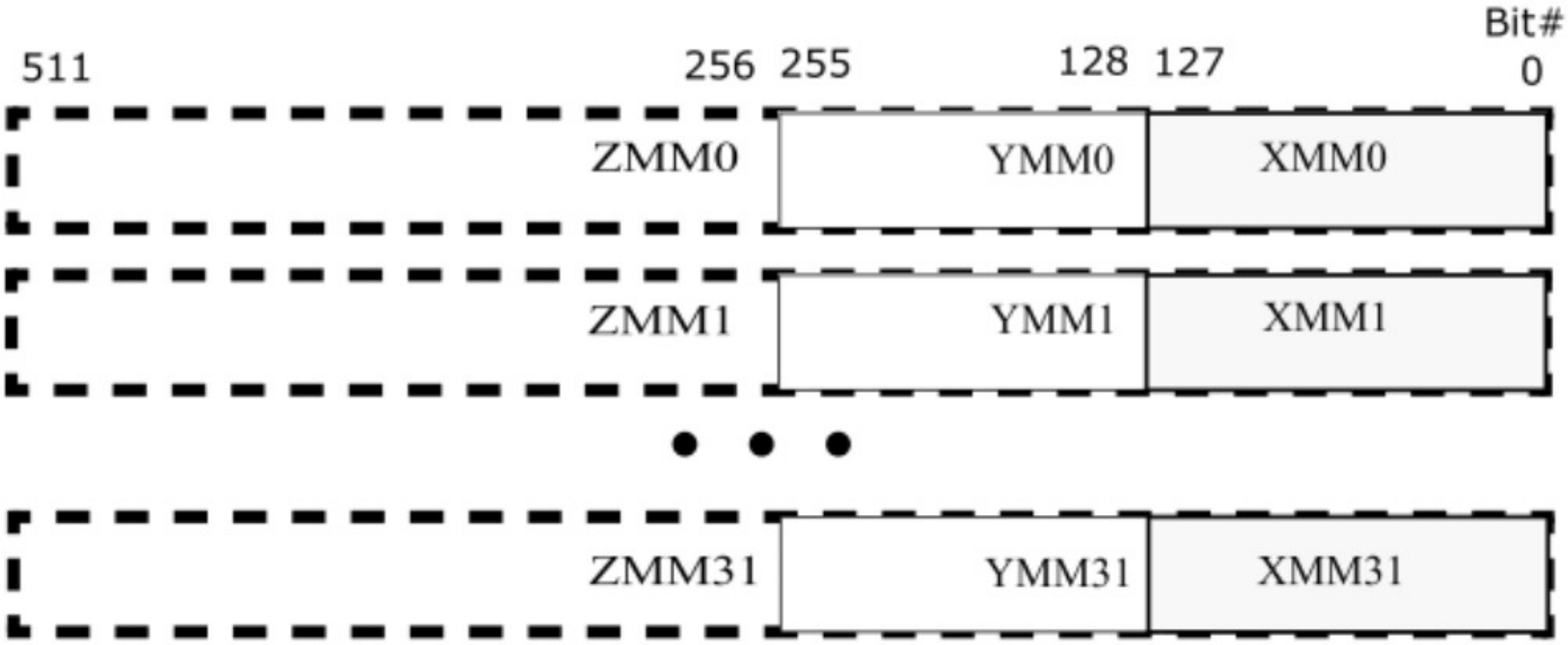


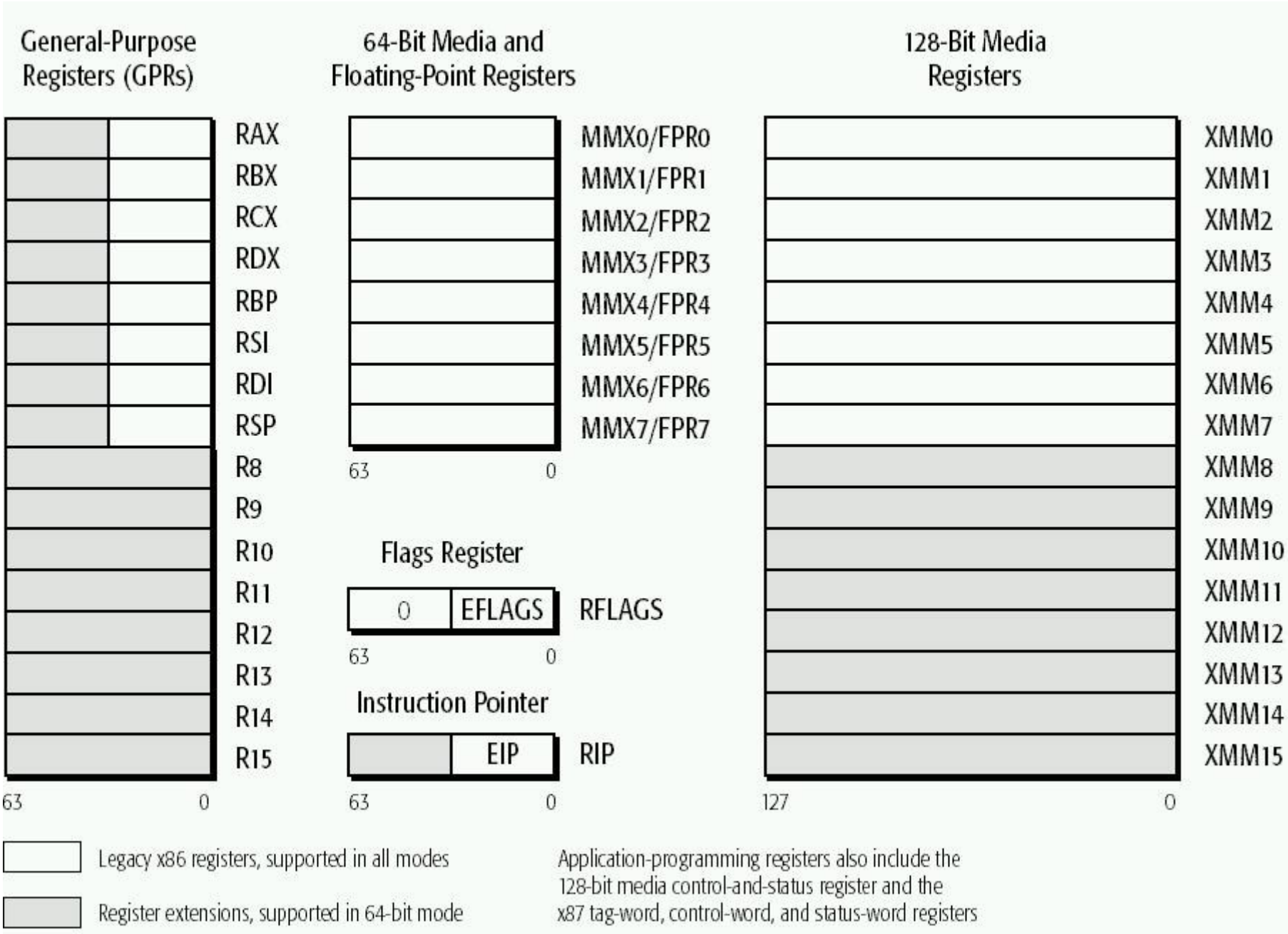
Beispiel aus der Grafikverarbeitung:

Nicht nur ein Punkt der ursprünglichen Grafik wird skaliert, gedreht und verschoben, sondern alle Punkte gleichzeitig.

XMM-, YMM und ZMM-Register

Einige der jüngeren x86-64 Prozessoren unterstützen auch größere Register.




General-Purpose Registers (GPRs)
64-Bit Media and Floating-Point Registers
128-Bit Media Registers
RAX
RBX
RCX
RDX
RBP
RSI
RDI
RSP
R8
R9
R10
R11
R12
R13
R14
R15
63
0
MMX0/FPR0
MMX1/FPR1
MMX2/FPR2
MMX3/FPR3
MMX4/FPR4
MMX5/FPR5
MMX6/FPR6
MMX7/FPR7
Flags Register
0
EFLAGS
RFLAGS
Instruction Pointer
EIP
RIP
XMM0
XMM1
XMM2
XMM3
XMM4
XMM5
XMM6
XMM7
XMM8
XMM9
XMM10
XMM11
XMM12
XMM13
XMM14
XMM15
127
Legacy x86 registers, supported in all modes
Register extensions, supported in 64-bit mode
Application-programming registers also include the 128-bit media control-and-status register and the x87 tag-word, control-word, and status-word registers