1 Base Components of a Computer
1.1 Von Neumann Architecture

„Father“ of most computers is the classic universal computing automaton

- Basic idea postulated in a concept paper by John v. Neumann

- Implemented as IAS-(Institute of Advanced Studies)-Rechner (Burks, Goldstine, John von Neumann at Princeton, 1946)

Despite huge technological changes, the basic principle is still found in modern microprocessors today
1 Base Components of a Computer
1.1 Von Neumann Architecture

Concept is based on four principles

1. Computer consists of 4 units

Control Unit is interpreting programs

Main Memory for program and data

Executes arithmetical and logical operations

I/O Units communicates with „outside world“

Also: Secondary Memory (e.g. hard disk)
2. Computer is independent of the problem it is supposed to solve: computer is controlled by programs.

### HW-Programming

- **Data**
- **Result**

- (fixed!) sequence of arithmetical and logical functions

### Controlled by program

- **Instructions**
- **Control Signals**
- **Data**
- **Results**

- Instruction Interpreter (Control Unit)
- General purpose arithmetical and logical functions

**Von Neumann Architecture**

- Computer is independent of the problem it is supposed to solve: computer is controlled by programs.
1 Base Components of a Computer
1.1 Von Neumann Architecture

3. Program and data are located in the same memory. In theory they can be modified by the computer (Note: it is not common to modify program data during program execution)

4. Main Memory is divided into cells of equal size and sequentially numbered (memory address). Data and instructions are referenced by memory addresses

5. A program consists of a sequence of instructions, which are executed sequentially per default
6. Exception from sequential execution of instructions by means of conditional and unconditional branch (jump) instructions

- Causes jump to the addressed cell in main memory
- Conditional branches depend on the result of arbitrary operations

7. The computer uses binary numbers
1 Base Components of a Computer
1.1 Components (1)

Control Unit
- Microprogramming
- ISA – Instruction Set Architectures
- Memory and Register Addressing

Compute Unit (also: Arithmetical Logical Unit – ALU)
- Integer and Floating Point Units

Memory Unit
- Memory Hierarchy
- Internal and External Memory Organization

I/O Unit
- Communication via Networks/Busses
1 Base Components of a Computer
1.1.1 Microprogramming (1)

Microprogram
- Mentioned for the first time by M.V. Wilkes (early 1950s)
- Core of an von Neumann Architecture
- Breakthrough 1964 (IBM System/360)

Why microprograms?
- Semantic gap
- Memory was rare and expensive in the early days
- Flexibility

Microprograms
- Sequence of micro instructions
1 Base Components of a Computer
1.1.1 Microprogramming (2)

Microprogram vs. computer program

- To avoid confusion with nomenclature

  - Programming of computer through Instructions
    - Assembly instructions (add, sub, mul), to distinguish them from microinstructions, we call them macroinstructions

  - Microprogramming of a computer via firmware
    - Firmware = Set of all Microprograms
    - Microprogram = Sequence of Microinstructions

  - One assembly/(macro)instruction is mapped onto a microprogram, which is made up of several microinstructions
1 Base Components of a Computer
1.1.1 Microprogramming (3)

Relation macroinstruction – microprogram/microinstruction

- Macroinstruction is an entry into a microprogram
  - A microprogram can also include branches

(Macro-) Instruction

Microprogram

Microinstr. 1

Microinstr. 2

Microinstr. 3

…

Microinstr. n
1.1.1 Microprogramming (4)

- Question: How is a (macro-)program mapped onto a semantically equivalent series of microprograms?
A control and a compute unit is required

<table>
<thead>
<tr>
<th>Control Unit</th>
<th>Load B</th>
<th>Load A</th>
<th>A + B</th>
<th>A - B</th>
<th>B - A</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Diagram:
- **Input** from Memory
- **From Memory**
- **Adder**
- **Output** to Memory
- **Select Input**
- **Select Memory**
- **Select Adder**
- **Select C**
- **Select D**
- **Select Output**
Execution of the following macroinstruction

\[ \text{A} - \text{B} \rightarrow \text{A} \]

Which microinstructions are necessary? Which control signals must be set?

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Load B</td>
<td>Load A</td>
<td>A+B</td>
<td>A-B</td>
<td>B-A</td>
<td>A</td>
<td>Load C</td>
<td>Select Adder</td>
<td>Select Input</td>
<td>Select Memory</td>
<td>Load D</td>
<td>Load E</td>
<td>Select C</td>
<td>Select D</td>
<td>Select Memory</td>
<td>Select Output</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Load A-B \rightarrow C; go to Microinstruction 2
Load C \rightarrow A; go to Microinstruction 3
What about branches?

- cf. Slide 10; Instruction 19
- Control unit has to be extended
- Address of next instruction in F
1. Base Components of a Computer
1.1.1 Microprogramming (8)

Extention in detail:

- $C < 0 \rightarrow \text{sign bit } C = '1'$
  $C \geq 0 \rightarrow \text{sign bit } C = '0'$

- if $(C < 0)$
  then $F = F + 1 + 1$
  else $F = F + 1 + 0$

- Equivalent to
  $F = F + 1 + \text{sign bit } C$

Address of next micro instruction
Micro instruction sequence with \textit{branch}


d| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17
---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---
1 | Load B | Load A | A+B | A-B | B-A | A | Load C | Select Adder | Select Input | Select Memory | Load D | Load E | Select C | Select D | Select Memory | Select Output | Jump, if C < 0
2 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0
3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1
4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0

A-B -> C and D; goto microinstruction 2 if C < 0; goto microinstruction 4
D -> E; goto microinstruction 4
...

(Vertical) coding of control signals

<table>
<thead>
<tr>
<th>Vertically coded Control Signals</th>
<th>Signals for Decoder</th>
<th>Decoder</th>
<th>Signals for Control Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load</td>
<td>000</td>
<td>000</td>
<td>1 Load B</td>
</tr>
<tr>
<td>Load</td>
<td>001</td>
<td>001</td>
<td>2 Load A</td>
</tr>
<tr>
<td>Load</td>
<td>010</td>
<td>010</td>
<td>3 Load C</td>
</tr>
<tr>
<td>Load</td>
<td>011</td>
<td>011</td>
<td>4 Load D</td>
</tr>
<tr>
<td>Load</td>
<td>100</td>
<td>100</td>
<td>5 Load E</td>
</tr>
<tr>
<td>Load</td>
<td>101</td>
<td>101</td>
<td>6 A</td>
</tr>
<tr>
<td>Load</td>
<td>110</td>
<td>110</td>
<td>7 A + B</td>
</tr>
<tr>
<td>Adder</td>
<td>00</td>
<td>00</td>
<td>8 Select Adder</td>
</tr>
<tr>
<td>Adder</td>
<td>01</td>
<td>01</td>
<td>9 Select Input</td>
</tr>
<tr>
<td>Adder</td>
<td>10</td>
<td>10</td>
<td>10 Select Memory</td>
</tr>
<tr>
<td>Adder</td>
<td>11</td>
<td>11</td>
<td>11 Select C</td>
</tr>
<tr>
<td>Memory</td>
<td>00</td>
<td>00</td>
<td>12 Select D</td>
</tr>
<tr>
<td>Memory</td>
<td>01</td>
<td>01</td>
<td>13 Jump, if C &lt; 0</td>
</tr>
<tr>
<td>Memory</td>
<td>10</td>
<td>10</td>
<td>14 Jump, if C &lt; 0</td>
</tr>
<tr>
<td>memory</td>
<td>11</td>
<td>11</td>
<td>15 Jump, if C &lt; 0</td>
</tr>
<tr>
<td>Output</td>
<td>00</td>
<td>00</td>
<td>16 Select Memory</td>
</tr>
<tr>
<td>Output</td>
<td>01</td>
<td>01</td>
<td>17 Select Memory</td>
</tr>
<tr>
<td>Output</td>
<td>10</td>
<td>10</td>
<td>18 Select Memory</td>
</tr>
<tr>
<td>Output</td>
<td>11</td>
<td>11</td>
<td>19 Select Memory</td>
</tr>
<tr>
<td>Select Register</td>
<td>00</td>
<td>00</td>
<td>20 Select Memory</td>
</tr>
<tr>
<td>Select Register</td>
<td>01</td>
<td>01</td>
<td>21 Select Memory</td>
</tr>
<tr>
<td>Select Register</td>
<td>10</td>
<td>10</td>
<td>22 Select Memory</td>
</tr>
<tr>
<td>Select Register</td>
<td>11</td>
<td>11</td>
<td>23 Select Memory</td>
</tr>
<tr>
<td>Jump</td>
<td>00</td>
<td>00</td>
<td>24 Select Memory</td>
</tr>
<tr>
<td>Jump</td>
<td>01</td>
<td>01</td>
<td>25 Select Memory</td>
</tr>
<tr>
<td>Jump</td>
<td>10</td>
<td>10</td>
<td>26 Select Memory</td>
</tr>
<tr>
<td>Jump</td>
<td>11</td>
<td>11</td>
<td>27 Select Memory</td>
</tr>
</tbody>
</table>
### Base Components of a Computer

#### 1.1.1 Microprogramming (11)

**Coding of Control Signals**

<table>
<thead>
<tr>
<th>Micro-Instructions</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Load</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Adder</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Adder</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **A-B -> C; go to microinstruction 2**
- **A-B -> D; go to microinstruction 3**
- **if C < 0; go to microinstruction 5**
- **D -> E; go to microinstruction 5**
- **End**
1 Base Components of a Computer
1.1.1 Microprogramming (12)

Structure of a micro-programmed architecture
- (CAR) Control Address Register: contains address of next instruction
- (CBR) Control Buffer Register: contains data from micro program
- Sequencing Logic
  - Generates Read Signals
  - Decides from where to take next address
1 Base Components of a Computer
1.1.1 Microprogramming (13)

Horizontal micro programming

- Horizontal micro instruction extended with branch address

- System Bus: connects all components
- Internal CPU signals: wires that directly connect to the ALU and instruct it to perform operations
- Indirect bit: indicates whether indirect addressing is used/not used
Vertical Microprogramming

- Vertical Microinstruction with branch address

- Instead of using control signals directly, a decoder, which creates the actual micro instructions, is used

- Advantage/Disadvantage
  - More compact representation at the cost of additional decoder logic
1. Base Components of a Computer

1.1.1 Microprogramming (15)

Overview of functions of a microprogrammed control unit

![Diagram of a microprogrammed control unit](image-url)
(External) Interrupt handling and indirect addressing

- Modify instruction cycle
  - Division in several states
    - Fetch, Execute, Indirect Addressing, Interrupt

Diagram:

```
+----------------+       +----------------+       +----------------+
| Fetch           |       | Execute         |       | Interrupt       |
|                 |       |                 |       |                 |
+----------------+       +----------------+       +----------------+
|                 |       |                 |       | Indirect        |
+----------------+       +----------------+       +----------------+
```

1 Base Components of a Computer
1.1.1 Microprogramming (16)
1 Base Components of a Computer
1.1.1 Microprogramming (17)

Instruction cycle (More detailed)

- Takes indirect addressing and interrupts into account
1 Base Components of a Computer
1.1.1 Microprogramming (18)

Control unit memory

- Fixed sequence of micro instruction, which implement the previously shown instruction cycle
  - Tasks „Instruction Fetch“, checking for „indirect addressing“ and „interrupt“ are implemented as micro programs
  - Fixed micro programs, that cannot be changed
  - Starting with AND micro program are the instructions that implement actual instruction
  - New micro programs (new macro instructions) can be added by means of firmware update
Summary: Advantages micro programming

- Micro program memory can be changed
  - Leads to higher flexibility

- Instruction (Set) Compatibility
  - New processor can „understand“ old instructions (can be implemented as different microprogram)

- Instruction Emulation
  - Processor can understand instruction set of another architecture

- Idea: Don’t use micro program memory for instruction „Translation“. Use software instead
  - Principal idea behind virtual machines
1 Base Components of a Computer

1.1.2 Interrupt Handling

Simplified interrupt handling

User (main) program is interrupted and continued by another program (interrupt handler) that is executed as consequence of the occurred interrupt.

After finishing this program, execution continues after the location where the interrupt took place.
1 Base Components of a Computer
1.1.2 Interrupt Handling

State before interrupt

Main Memory

Address

Program

Interrupt-Handler

Data

Stack

CPU

Register

A 0815
B 3141
C 4711
D 2718

Program Counter

PC 73

Components of a Computer

State before interrupt
1.1.2 Interrupt Handling

1. Incoming interrupt

2. Back up PC onto Stack

3. Load address of first instruction of interrupt handler to PC

Interrupt-Id = 158 = address for instruction of interrupt routine
1. Base Components of a Computer
1.1.2 Interrupt Handling

Address of Interrupt Routine loaded into PC

CPU

Register

A
B
C
D

Program Counter

PC

Main Memory

Adresse

Program

Interrupt-Handler

Data

Stack

Adresse

158

158

73

Program Memory

Addresses

158

158

73

Base Components of a Computer
1.1.2 Interrupt Handling

Address of Interrupt Routine loaded into PC

CPU

Register

A
B
C
D

Program Counter

PC

Main Memory

Adresse

Program

Interrupt-Handler

Data

Stack

Adresse

158

158

73

Program Memory

Addresses
4. Back up contents of CPU registers onto stack

push A
push B
push C
push D

CPU

Register
A 0815
B 3141
C 4711
D 2718

Program Counter
PC 158

Main Memory

Address
Program
Interrupt-Handler
Data
Stack

Interrupt Handling
1. Base Components of a Computer

1.1.2 Interrupt Handling

5. Complete Interrupt Routine

CPU

Register

A
B
C
D

Program Counter

PC

Main Memory

Address

Program

Interrupt-Handler

Data

Stack

xxxx
xxxx
xxxx
2718
4711
3141
...

Base Components of a Computer

1.1.2 Interrupt Handling
1 Base Components of a Computer

1.1.2 Interrupt Handling

CPU

- pop D
- pop C
- pop B
- pop A
- pop PC

Register

- A
- B
- C
- D

Program Counter

PC

Main Memory

- Program
- Interrupt-Handler
- Data
- Stack

0815
3141
4711
2718
73
0815
2718
4711
3141
73
1 Base Components of a Computer

1.1.2 Interrupt Handling

State is now the same as before interrupt

Main Memory

Address

Program

Interrupt-Handler

Data

Stack

CPU

Register

A 0815
B 3141
C 4711
D 2718

Befehlszähler

PC 73

CPU

State is now the same as before interrupt

Main Memory

Address

Program

Interrupt-Handler

Data

Stack

CPU

Register

A 0815
B 3141
C 4711
D 2718

Befehlszähler

PC 73
1. Base Components of a Computer

1.1.2 Interrupt Handling

7. Increment PC and continue program

Main Memory

Address

Program Data

Interrupt-Handler

Data

Stack

CPU

Register

A  0815
B  3141
C  4711
D  2718

Program Counter

PC  74

1 Base Components of a Computer

1.3 Memory (1)

Apart from the actual compute unit, the memory unit in a computer is crucial in terms of

- Performance and
- Cost of a computer

Ideally

- Sufficient capacity
- Memory access time can keep up with processing speed of the compute unit
- Can’t be realized due to economic and technological reasons

Solution: Memory hierarchy

- Each memory level is smaller, faster, and more expensive (per byte) than the next level
- (Condition of inclusion: every memory in the hierarchy contains part of the data from the next, higher hierarchy stage)
1.3.1 Memory Hierarchy (1)

- Memory hierarchy and properties (approximate)

<table>
<thead>
<tr>
<th>Memory</th>
<th>Acces time</th>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor register</td>
<td>1 clock cycle</td>
<td>256 - 1024 Bytes</td>
</tr>
<tr>
<td>Primary cache (40:1 zu HS)</td>
<td>1-4 clock cycles</td>
<td>1 - 128 KBytes</td>
</tr>
<tr>
<td>Secondary cache (10:1 to main memory)</td>
<td>3-10 clock cycles</td>
<td>256 KB - 4 MBytes</td>
</tr>
<tr>
<td>Main memory</td>
<td>25-266 clock cycles</td>
<td>~ GBytes</td>
</tr>
<tr>
<td>Background memory</td>
<td>5 - 15 ms</td>
<td>up to 5 TByte</td>
</tr>
<tr>
<td>Archive memory</td>
<td>&gt;&gt; 50 ms</td>
<td>several TBytes</td>
</tr>
</tbody>
</table>

- Caching of data
  - From main memory and registers in Cache by HW
  - From background memory to main memory by SW (operating system)
• Visualization of the **condition of inclusion** within the memory hierarchy

![Diagram of Memory Hierarchy](image)

- **Chassis**
  - Processor
  - Register
  - On-chip cache
  - Primary Cache (1st level cache)
  - Secondary and tertiary caches (2nd, 3rd level caches)
  - 2nd & 3rd level cache (SRAM)
  - Main Memory (DRAM)
  - Main Memory (DRAM)

- **External Memory**
  - Magnedic, solid state, optical drives
  - e.g. magnetic tapes

1 Base Components of a Computer
1.3.1 Memory Hierarchy (2)
1. Base Components of a Computer
1.3.1 Memory Hierarchy (3)

Layout of memory

<table>
<thead>
<tr>
<th>Capacity</th>
<th>Speed</th>
<th>Price (€/bit)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Diagram showing the hierarchy of memory:
- Inboard memory
- Registers Cache Main memory
- Outboard storage
- Magnetic disk CD-ROM CD-RW DVD+RW DVD-RAM
- Off-Line storage
- Magnetic tape MO WORM
## Properties of memory types

<table>
<thead>
<tr>
<th>Location</th>
<th>Performance</th>
<th>Physical Type</th>
<th>Physical Characteristics</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Access time</td>
<td>Semiconductor</td>
<td>Volatile/nonvolatile</td>
<td></td>
</tr>
<tr>
<td>Internal (main)</td>
<td>Cycle time</td>
<td>Magnetic</td>
<td>Erasable/nonerasable</td>
<td></td>
</tr>
<tr>
<td>External (secondary)</td>
<td>Transfer rate</td>
<td>Optical</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Magneto-optical</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Capacity</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Word size</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of words</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Unit of Transfer</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Word</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Access Method</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequential</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Direct</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Associative</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1 Base Components of a Computer
1.3.2 Memory Structure (1)

Layout of Memory System

- Chipset (Explained later, c.f. Slide 65)
Processor mostly accesses fast memory (caches)

- Possible due to the **temporal and spatial locality** of data and instructions
Cache

- Small and fast buffer memory between register file and main memory

Purpose of a cache

- Bridge the gap in performance of processor and main memory
  - Processor computes data much faster than main memory delivers it

Cache organized in blocks (just like main memory)

- Mostly cache lines of size 64B (Intel, AMD) or 128B (IBM POWER)
Main memory

- Today typically made from DRAM (dynamic RAM)
- Caches are typically built from SRAM (static RAM)

SRAM chips

- Memory cell: Flip-Flop
  - Non-destructive reading possible
  - Bigger than DRAM cell: 6-8 Transistors
  - faster: ~ factor 8
  - Lower capacity: ~ factor 8
1 Base Components of a Computer
1.4 Main Memory Design

**DRAM-Chips**

- Advantage: very compact
- Disadvantage: Reading data is destructive; Line has to be written back by read-/write-amplifier

- Dynamic RAM Memory Cell: Transistor plus Capacitor

```
+-----------------+    +-----------------+
|                 |    |                 |
|     Bit Line    |    |     Capacitor   |
|-----------------+    |-----------------+
|                 |    |                 |
+-----------------+    |     Transistor   |
                   +-----------------+
```

Address Line
1 Base Components of a Computer
1.4 Main Memory Design

- Main memory is made up of memory matrices

- Memory matrix
  with one or multiple 1-bit cells at nodes

- Addressing via row and column multiplexing
  - Read whole line via row address
  - Addressing of bit/bits via column address

- Due to leakage currents, every line has to be refreshed (rewritten) at intervals of approximately 64 ms
  - This is done by reading (and an implicit write back) of the line's contents
  - Either block-wise, i.e. all line entries are refreshed simultaneously; or
  - Entries are refreshed individually using a fixed address pattern between the regular memory accesses
Schematic overview of a 4M×1-Bit DRAM

- Without refreshing logic
1. Base Components of a Computer
1.4 Main Memory Design

- Example: Byte-addressable 16 MByte memory with 32-Bit words from 4M×1-Bit DRAMs

- Memory Controller is responsible for
  - Address interpretation
  - Word addressing and
  - Selection of one or more byte blocks

- Memory Bank: Memory controller + multiple memory blocks arranged in parallel
Memory interleaving

- Cycle time slowing down processor
  - Example: 1 GHz processor and memory cycle time of 20 ns → Processor can access memory only every 20 clock cycles

- Solution: Memory interleaving
  - Adjacent words are located in different memory banks
  - Memory access to different memory banks can overlap

- Example: 4-way interleaved memory
Recent developments in technology

- Latency of memory saw a yearly decrease of only 10%
  - This is known as the memory gap
1 Base Components of a Computer
1.4 Main Memory Design

- To decrease latency, improvements in memory design were necessary
  - Nibble-, Page- or Static Column-Mode: reading of multiple consecutive bits in the active line at each memory access
    - e.g. EDO-RAM
  - EDRAM (enhanced DRAM) or CDRAM (cached DRAM)
    - Cache integrated on memory chip
  - SDRAM (synchronous DRAM)
    - Is operated synchronous to processor- / memory bus
    - In addition: more memory matrices → memory interleaving (to enable Burst Mode: fast transmission of whole blocks)
      - At 100 MHz 10 ns for successive accesses
  - DDR (double data rate) – RAM
    - Data transmission at rising and falling edge
1 Base Components of a Computer
1.4 Main Memory Design

- Burst Access in SDRAM
  (in detail: Stallings, Chap. 5, 191-195, Synchronous DRAM)

  - Read \( n \) (four in the example) consecutive addresses at once
    - Latency: 2 cycles
    - Burst Length: \( n = 4 \)

```
<table>
<thead>
<tr>
<th>T0</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
<th>T8</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

COMMAND
- READ A
- NOP
- NOP
- NOP
- NOP
- NOP
- NOP
- NOP

DQs
- DOUT A₀
- DOUT A₁
- DOUT A₂
- DOUT A₃
```
1 Base Components of a Computer

1.4 Main Memory Design

- Burst access and alternating access to different memory banks
  [http://www.hardwareecke.de/berichte/grundlagen/dram_1.php](http://www.hardwareecke.de/berichte/grundlagen/dram_1.php)
  - Access memory banks out of phase (Precharge, Active, Write/Read)
1 Base Components of a Computer
1.4 Main Memory Design

- Today DDR technology is state of the art
- Read data by rising and falling edge

Comparison of DDR-Technologies

- SDRAM – 1 Word synchronous to bus frequency

- DDR1-400 – 2 Words
  Prefetch during rising and falling edge:
  two successive addresses are read

- DDR2-533
  Four words: rising and falling edge, doubles I/O frequency
### Technology overview [Source: Wikipedia]

<table>
<thead>
<tr>
<th>Chip</th>
<th>Module</th>
<th>Memory Frequency</th>
<th>I/O-Frequency $^2$</th>
<th>Effective Frequency $^3$</th>
<th>Bandwidth per Module</th>
<th>Dual-Channel Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR-200</td>
<td>PC-1600</td>
<td>100 MHz</td>
<td>100 MHz</td>
<td>200 MHz</td>
<td>1,6 GB/s</td>
<td>3,2 GB/s</td>
</tr>
<tr>
<td>DDR-266</td>
<td>PC-2100</td>
<td>133 MHz</td>
<td>133 MHz</td>
<td>266 MHz</td>
<td>2,1 GB/s</td>
<td>4,2 GB/s</td>
</tr>
<tr>
<td>DDR-333</td>
<td>PC-2700</td>
<td>166 MHz</td>
<td>166 MHz</td>
<td>333 MHz</td>
<td>2,7 GB/s</td>
<td>5,4 GB/s</td>
</tr>
<tr>
<td>DDR-400</td>
<td>PC-3200</td>
<td>200 MHz</td>
<td>200 MHz</td>
<td>400 MHz</td>
<td>3,2 GB/s</td>
<td>6,4 GB/s</td>
</tr>
</tbody>
</table>

$^2$ Frequency of bus between memory controller and RAM
$^3$ Doubled effective frequency due to double accesses at rising and falling clock edge

<table>
<thead>
<tr>
<th>Chip</th>
<th>Module</th>
<th>Memory Frequency</th>
<th>I/O-Frequency $^2$</th>
<th>Effective Frequency $^3$</th>
<th>Bandwidth per Module</th>
<th>Dual-Channel Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR2-400</td>
<td>PC2-3200</td>
<td>100 MHz</td>
<td>200 MHz</td>
<td>400 MHz</td>
<td>3,2 GB/s</td>
<td>6,4 GB/s</td>
</tr>
<tr>
<td>DDR2-533</td>
<td>PC2-4200</td>
<td>133 MHz</td>
<td>266 MHz</td>
<td>533 MHz</td>
<td>4,2 GB/s</td>
<td>8,4 GB/s</td>
</tr>
<tr>
<td>DDR2-667</td>
<td>PC2-5300</td>
<td>166 MHz</td>
<td>333 MHz</td>
<td>667 MHz</td>
<td>5,3 GB/s</td>
<td>10,6 GB/s</td>
</tr>
<tr>
<td>DDR2-800</td>
<td>PC2-6400</td>
<td>200 MHz</td>
<td>400 MHz</td>
<td>800 MHz</td>
<td>6,4 GB/s</td>
<td>12,8 GB/s</td>
</tr>
<tr>
<td>DDR2-1066</td>
<td>PC2-8500</td>
<td>266 MHz</td>
<td>533 MHz</td>
<td>1066 MHz</td>
<td>8,5 GB/s</td>
<td>17,0 GB/s</td>
</tr>
</tbody>
</table>
1 Base Components of a Computer
1.5 I/O-Logic: Introduction to the bus

Schematic Overview of the bus in a PC

- Connects Periphery, Memory, CPU together

Spatial Classification

- Important: Bus-Arbiter (not shown) manages communication of components on bus

Functional Classification
1 Base Components of a Computer
1.5 I/O-Logic: Introduction to the bus

Alternative to Bus?

- Bus: in principle, all components are connected via a single line
- Network with Point-to-Point (P2P) connections and router nodes
1 Base Components of a Computer
1.5.1 PCI-Bus

PCI-Bus (Peripheral Component Interface)
- Developed as bus system by Intel for the Pentium processor
- Bus frequency synchronous to CPU frequency (max. 33/66MHz)
- Bus width of 32 or 64 bit
  - For 32-bit bus maximum bandwidth of 44MB/s (read) or 66MB/s (write)

- Capable Burst-Mode
  - Peak bandwidth increases to 133 MB/s (32-bit bus) or 266 MB/s (64-bit bus)

- Use of bridges (in principle a chipset for the PCI-Bus)
  - To connect the PCI-BUS to other bus systems (e.g. PCI-to-ISA-Bridge)
1. Base Components of a Computer
1.5.1 PCI-Bus

- **Bus-Master- and Slave-Principle**
  - A PCI-Master can read/write data from/to main memory without using the CPU: Direct Memory Access (DMA)-Principle
  - A Slave on the other side can only be a receiver (e.g. PCI Graphics Card)
  - Automatic configuration of PCI cards
    - Configuration via ROM-BIOS
    - If there is a conflict, BIOS changes interrupt channels of hardware or disables hardware in case of errors
1 Base Components of a Computer

1.5.1 PCI-Bus

- Hierarchical PCI-System bus
  - PCI-Agents
    - PCI-Components
  - ISA/EISA-Bus
    - Predecessor of PCI
    - In (very) old machines: used to connect slow hardware
  - PCI Bridge
    - Supports Burst-Mode
## Base Components of a Computer

### 1.5.1 PCI-Bus

**Development PCI Bus**

<table>
<thead>
<tr>
<th>PCI-Version</th>
<th>PCI 2.0</th>
<th>PCI 2.1</th>
<th>PCI 2.2</th>
<th>PCI 2.3 / PCI 3.0</th>
<th>PCI-X-1.0</th>
<th>PCI-X-2.0</th>
<th>PCI-X-3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. Busbreite (Bit)</td>
<td>32</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>Max. Taktrate (MHz)</td>
<td>33</td>
<td>66</td>
<td>66</td>
<td>66</td>
<td>133</td>
<td>533</td>
<td>1066</td>
</tr>
<tr>
<td>Max. Bandbreite (GByte/s)</td>
<td>0,12</td>
<td>0,5</td>
<td>0,5</td>
<td>0,5</td>
<td>0,99</td>
<td>3,97</td>
<td>7,95</td>
</tr>
<tr>
<td>Slots pro Bridge</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Spannung (Volt)</td>
<td>5</td>
<td>5/3,3</td>
<td>5/3,3</td>
<td>3,3</td>
<td>3,3</td>
<td>3,3/1,5</td>
<td>3,3/1,5</td>
</tr>
</tbody>
</table>
1 Base Components of a Computer
1.5.2 PCIe-Bus

State of the Art – PCI-Express

- PCI-X bus a thing of the past
- PCIe: New faster serial connections

„Wir stehen vor dem radikalsten Redesign der PC-Plattform seit Einführung des PCI-Busses in den frühen 90er Jahren.“

P. Glasowsky, Microprocessor Report

From PCI over PCI-X to PCI-Express

- Higher Frequency
- Faster memory and slower I/O-Controller connected via data (memory) buffer called *bridge*
- PCI-X had a large number of (slow) lines (82)
- PCI Express: fewer, but faster lines (P2P)
1.5.2 PCIe-Bus

Difference PCI-(X) – PCI-Express
1 Base Components of a Computer
1.5.2 PCIe-Bus

Architecture – PCI-Express

- Root, Switches
- Links, Bridges

- 2.5-80 GBit/s depending on number of multiplexed lines
1 Base Components of a Computer

1.5.3 USB-Bus

USB (Universal Serial Bus)-Bus

- **Star-Topology**
  - Up to 127 devices

- **USB-Controller is in charge**
  - No direct connection between individual USB devices
  - USB Controller works as host
  - USB Controller is the only device that has to be assigned an interrupt line by the BIOS

- **Development**
  - In the beginning: 1.5 Mbit/s (USB 1.0) resp. 12 Mbit/s (USB 1.1)
  - USB 2.0 with max. 480 MBit/s
  - USB 3.0: 5.0 Gbit/s

- „hot-plugging“
  - Devices can be plugged in, reconnected while the computer is operating
1 Base Components of a Computer
1.5.3 Chipsets

Chipsets realise the interface of the Processor-Memory-Periphery System
Three fundamental modes of operation

Programmed Input/Output
- Handled by the processor

Interrupt-Driven I/O
- Handled by the processor when an interrupt occurs

DMA (Direct Memory Access)
- Only initiated by the processor
- DMA controller handles data transfer between main memory and periphery without the processor
1 Base Components of a Computer
1.6 Coupling of Memory with I/O
1 Base Components of a Computer
1.6.1 Addressing of Peripheral Devices

- Two addressing modes for peripheral devices
  - Mapped Memory
    - Periphery is addressed via normal memory (the devices’ control registers are mapped onto addresses in main memory)
  - Isolated I/O
    - Periphery is addressed by separated instructions (in, out vs. mov)

(a) Example „Memory-mapped I/O“

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>INSTRUCTION OPERAND</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Load AC „1“</td>
<td>Load accumulator</td>
</tr>
<tr>
<td>201</td>
<td>Store AC 517</td>
<td>Initiate keyboard read</td>
</tr>
<tr>
<td>202</td>
<td>Load AC 517</td>
<td>Get status byte</td>
</tr>
<tr>
<td>203</td>
<td>Branch if sign = 0</td>
<td>Loop until ready</td>
</tr>
<tr>
<td>204</td>
<td>Load AC 516</td>
<td>Load data byte</td>
</tr>
</tbody>
</table>

(b) Example „Isolated I/O“

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>INSTRUCTION OPERAND</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Load I/O 5</td>
<td>Initiate keyboard read</td>
</tr>
<tr>
<td>201</td>
<td>Test I/O 5</td>
<td>Check for completion</td>
</tr>
<tr>
<td>202</td>
<td>Branch Not Ready 201</td>
<td>Loop until ready</td>
</tr>
<tr>
<td>204</td>
<td>In 5</td>
<td>Load data byte</td>
</tr>
</tbody>
</table>
Base Components of a Computer

1.6.2 Interrupt driven I/O

Control flow: Interrupt driven I/O

Hardware

Device controller or other system hardware issues an interrupt

Processor finishes execution of current instruction

Processor signals acknowledgment of interrupt

Processor pushes PSW and PC onto control stack

Processor loads new PC value based on interrupt

Software

PSW: Program status word

PC: Program counter

Save remainder of process state information

Process interrupt

Restore process state information

Restore old PSW and PC
Control flow: programmed I/O (PIO)
- CPU takes care of everything
1.6.4 Interrupt driven I/O vs. DMA

**Interrupt driven I/O**
- CPU is waiting for interrupt
- Fetches data itself

**DMA**
- CPU issues job to the DMA controller
- DMA controller completes memory transfer
- CPU is notified of completion via interrupt by the DMA controller