Prof. Dr. Florian Künzner
Technical University of Applied Sciences Rosenheim, Computer Science

CA 5 – Processor 2

The lecture is based on the work and the documents of Prof. Dr. Theodor Tempelmeier

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Goal

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Goal

CA::Processor 2

- Endianness
- Examples
- Usage
- Transfer
- Solutions

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Endianness

Endianness: The definition of the **byte order** within a **multi-byte word**.

Register view of a 32bit architecture:

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Endianness

Endianness: The definition of the **byte order** within a **multi-byte word**.

Register view of a 32bit architecture:

CAMPUS Rosenheim
Computer Science

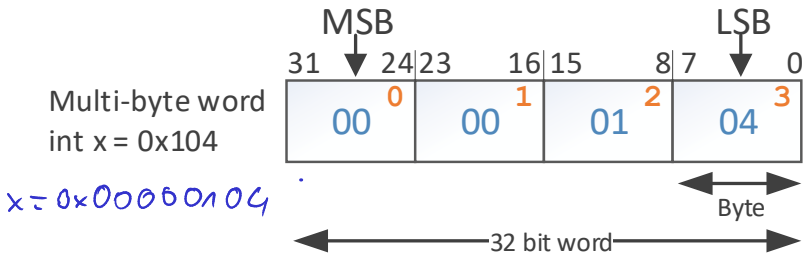
Technische Hochschule Rosenheim
Technical University of Applied Sciences

Endianness

Endianness: The definition of the **byte order** within a **multi-byte word**.

Register view of a 32bit architecture:

Big Endian



x = 0x00000104

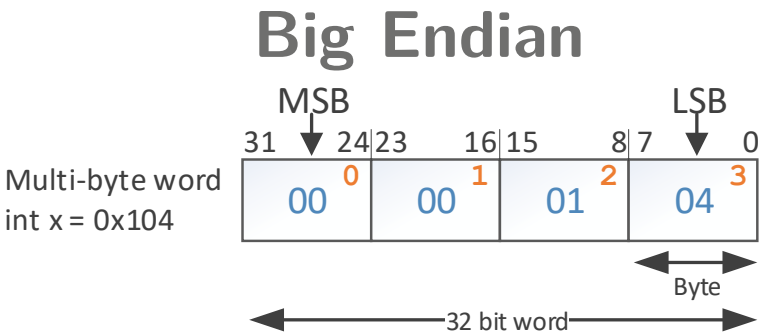
CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Endianness

Endianness: The definition of the **byte order** within a **multi-byte word**.

Register view of a 32bit architecture:



- MSB - Most significant byte
- LSB - Least significant byte

CAMPUS Rosenheim
Computer Science

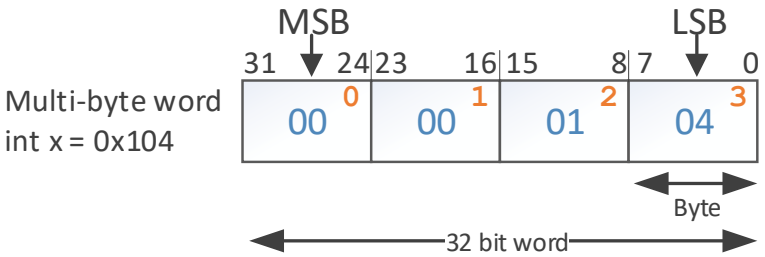
Technische Hochschule Rosenheim
Technical University of Applied Sciences

Endianness

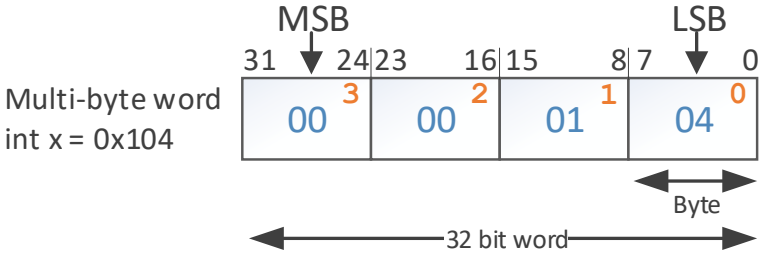
Endianness: The definition of the **byte order** within a **multi-byte word**.

Register view of a 32bit architecture:

Big Endian



Little Endian

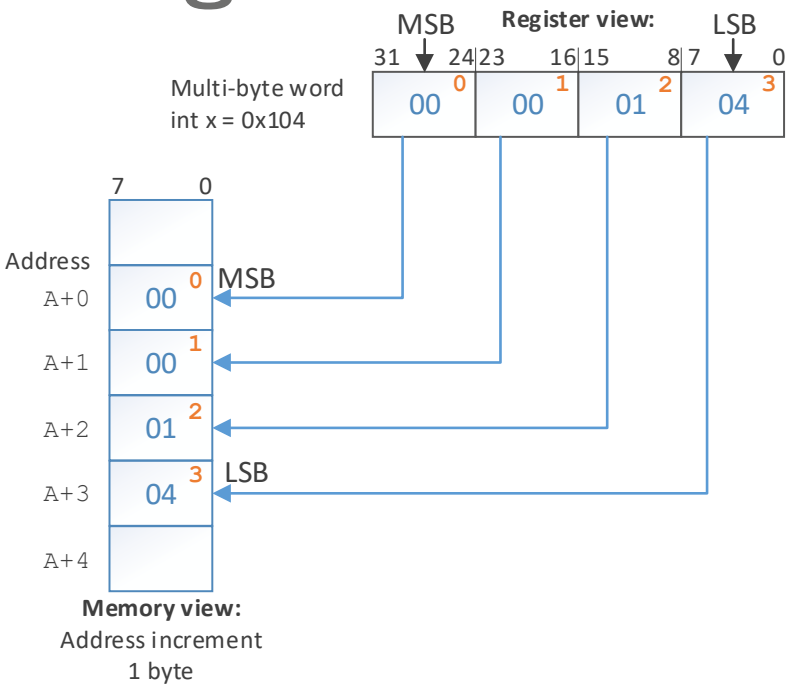


- MSB - Most significant byte
- LSB - Least significant byte

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

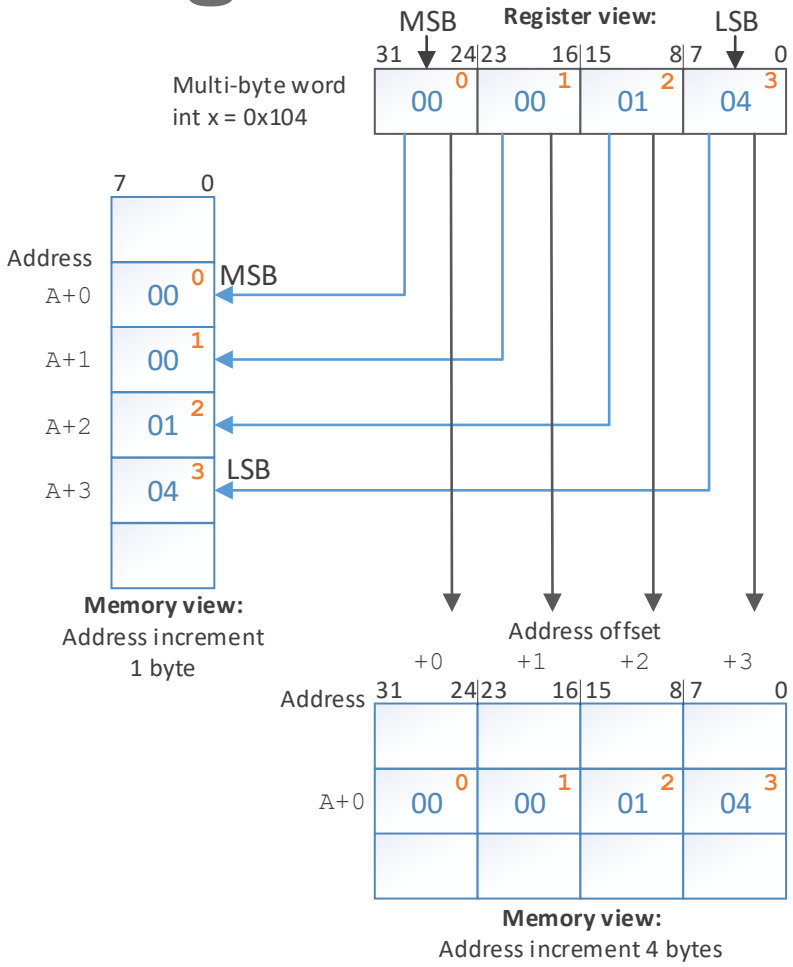
Endianness - Big endian (first)



CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

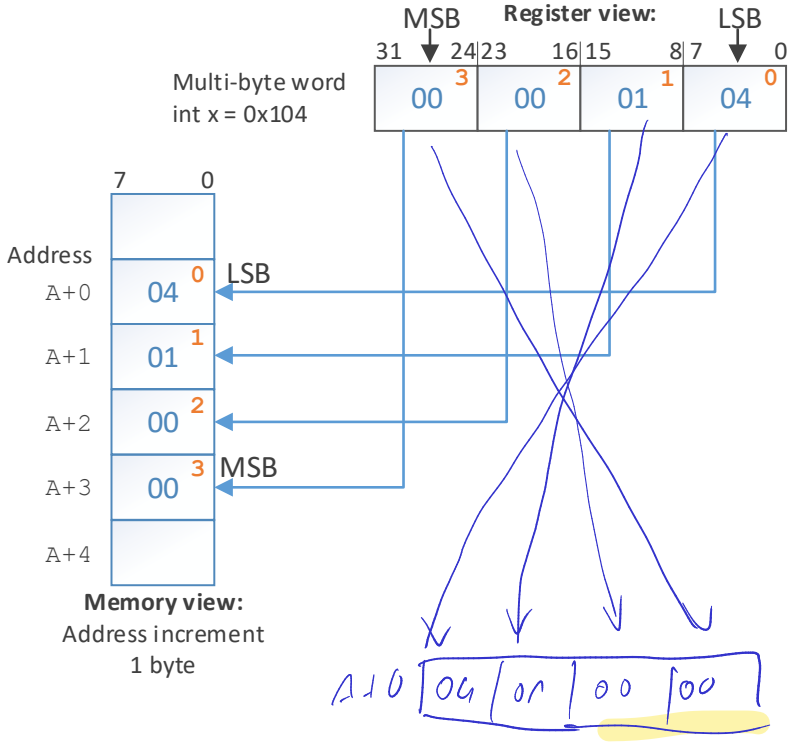
Endianness - Big endian



CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

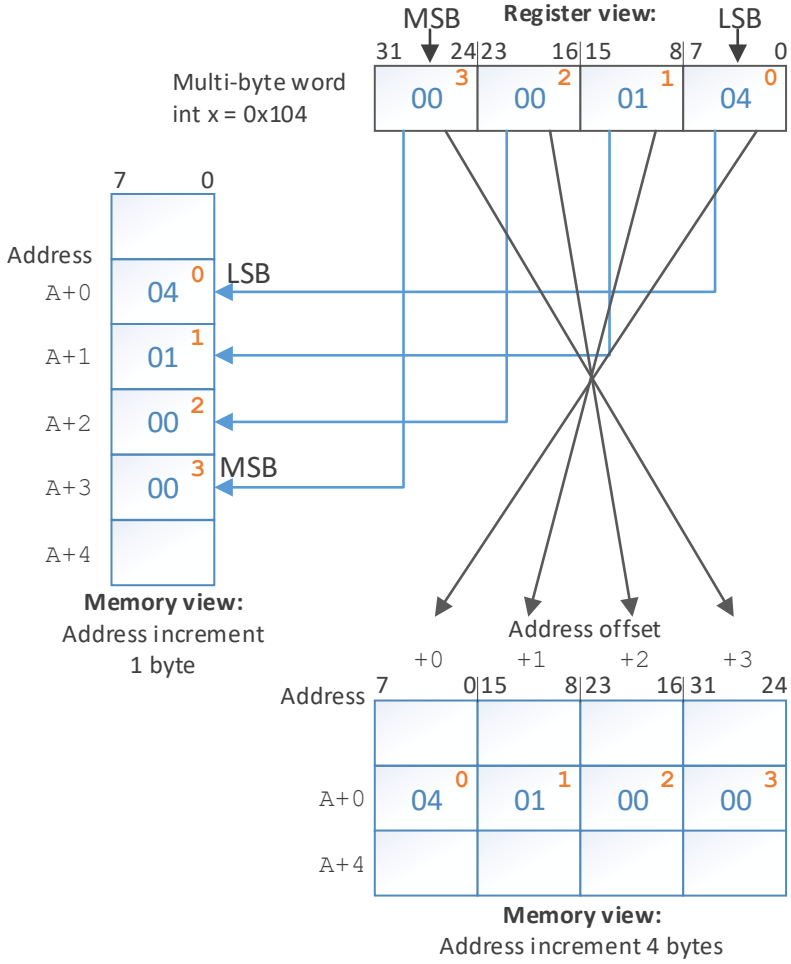
Endianness - Little endian (first)



CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Endianness - Little endian

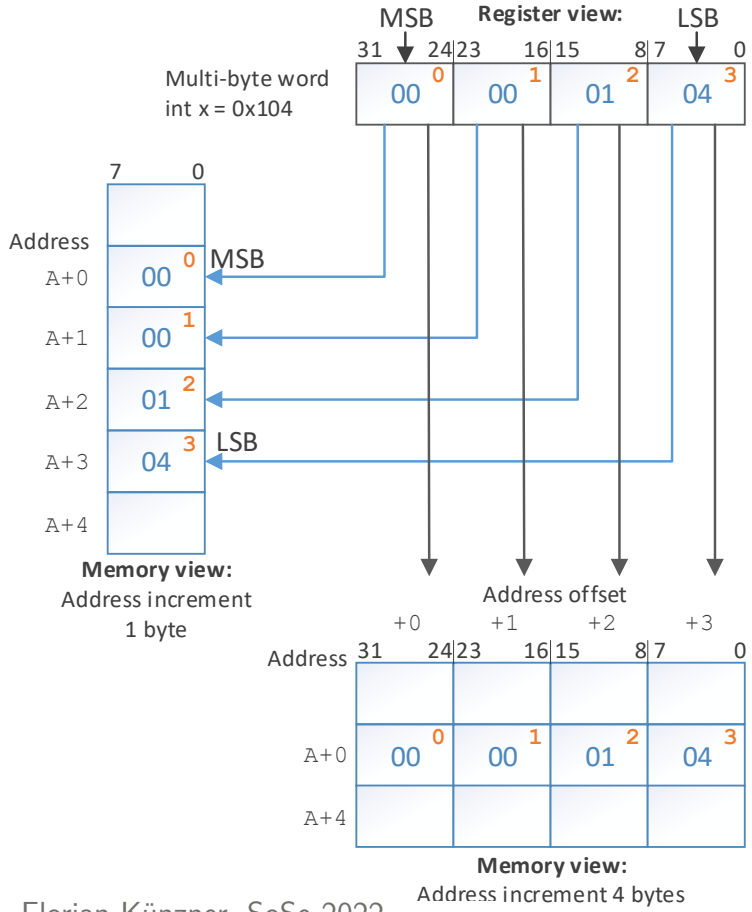


CAMPUS Rosenheim
Computer Science

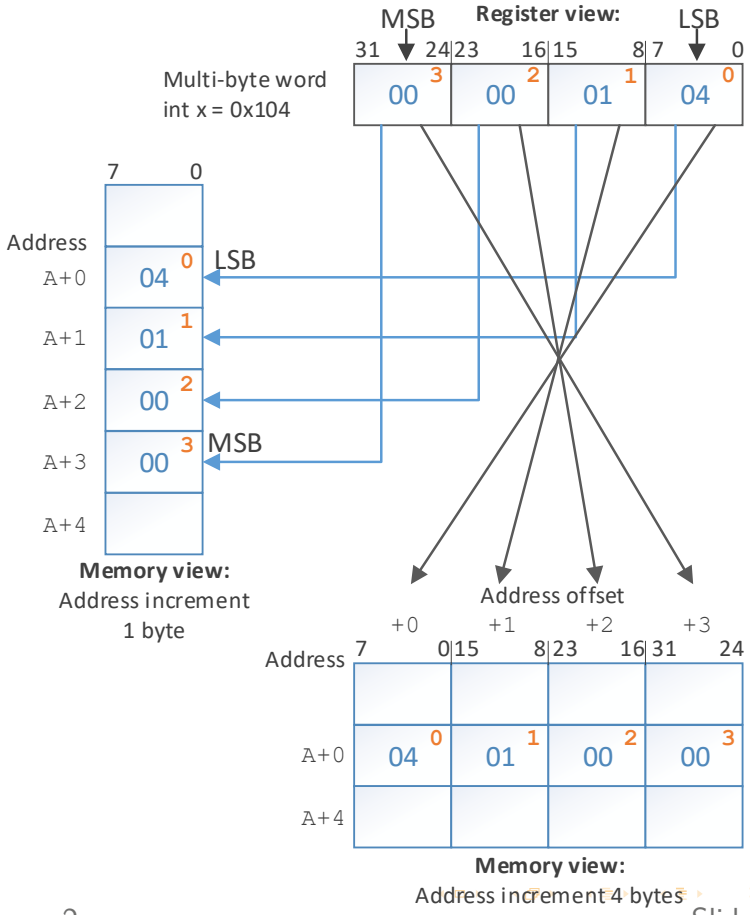
Technische Hochschule Rosenheim
Technical University of Applied Sciences

Endianness - BE/LE

Big endian



Little endian



CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Endianness - example BE

```
1  #include <stdlib.h>
2  #include <stdint.h>
3
4  int main()
5  {
6      struct employee {
7          char     name[12];
8          uint32_t age;
9          uint32_t dept_nr;
10     };
11
12     struct employee smith = {
13         .name    = "JIM SMITH",
14         .age     = 21,   //0x15
15         .dept_nr = 0x104 //260
16     };
17
18     return EXIT_SUCCESS;
19 }
```

[cmp: [1, p. 95-96]]

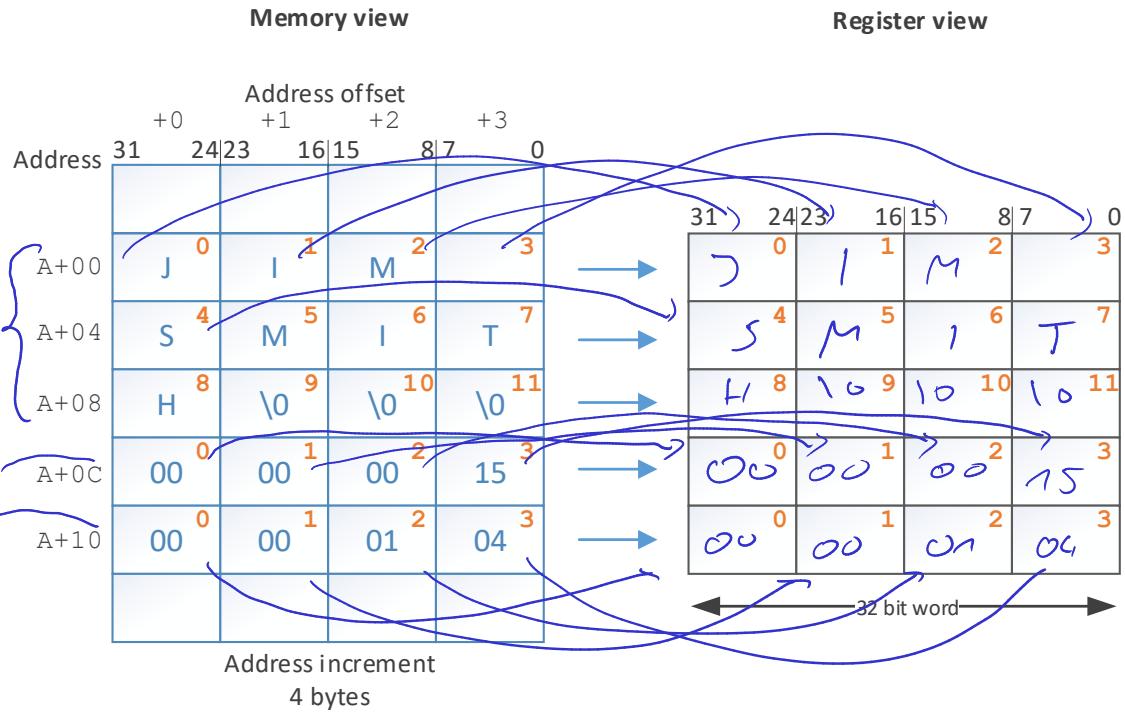
CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Endianness - example BE

Big endian memory -> Register

```
1  #include <stdlib.h>
2  #include <stdint.h>
3
4  int main()
5  {
6      struct employee {
7          char     name[12];
8          uint32_t age;
9          uint32_t dept_nr;
10     };
11
12     struct employee smith = {
13         .name    = "JIM SMITH",
14         .age     = 21,    //0x15
15         .dept_nr = 0x104 //260
16     };
17
18     return EXIT_SUCCESS;
19 }
```



[cmp: [1, p. 95-96]]

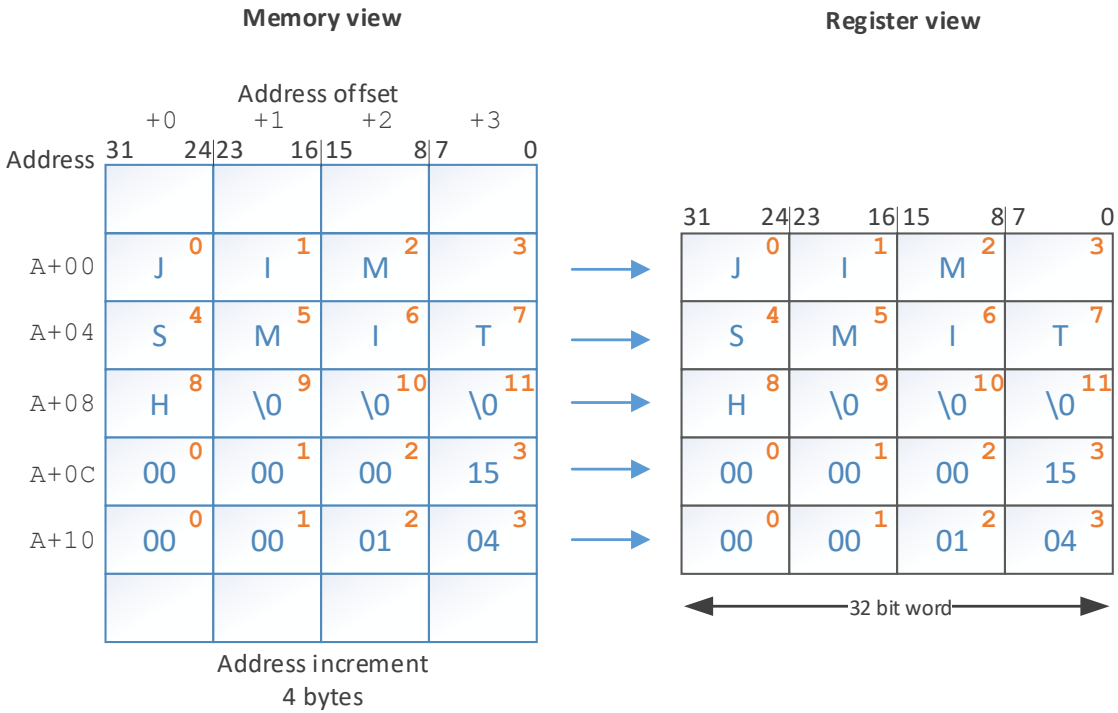
CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Endianness - example BE

Big endian memory -> Register

```
1  #include <stdlib.h>
2  #include <stdint.h>
3
4  int main()
5  {
6      struct employee {
7          char     name[12];
8          uint32_t age;
9          uint32_t dept_nr;
10     };
11
12     struct employee smith = {
13         .name    = "JIM SMITH",
14         .age     = 21,    //0x15
15         .dept_nr = 0x104 //260
16     };
17
18     return EXIT_SUCCESS;
19 }
```



[cmp: [1, p. 95-96]]

Endianness - example LE

```
1  #include <stdlib.h>
2  #include <stdint.h>
3
4  int main()
5  {
6      struct employee {
7          char     name[12];
8          uint32_t age;
9          uint32_t dept_nr;
10     };
11
12     struct employee smith = {
13         .name    = "JIM SMITH",
14         .age     = 21,   //0x15
15         .dept_nr = 0x104 //260
16     };
17
18     return EXIT_SUCCESS;
19 }
```

[cmp: [1, p. 95-96]]

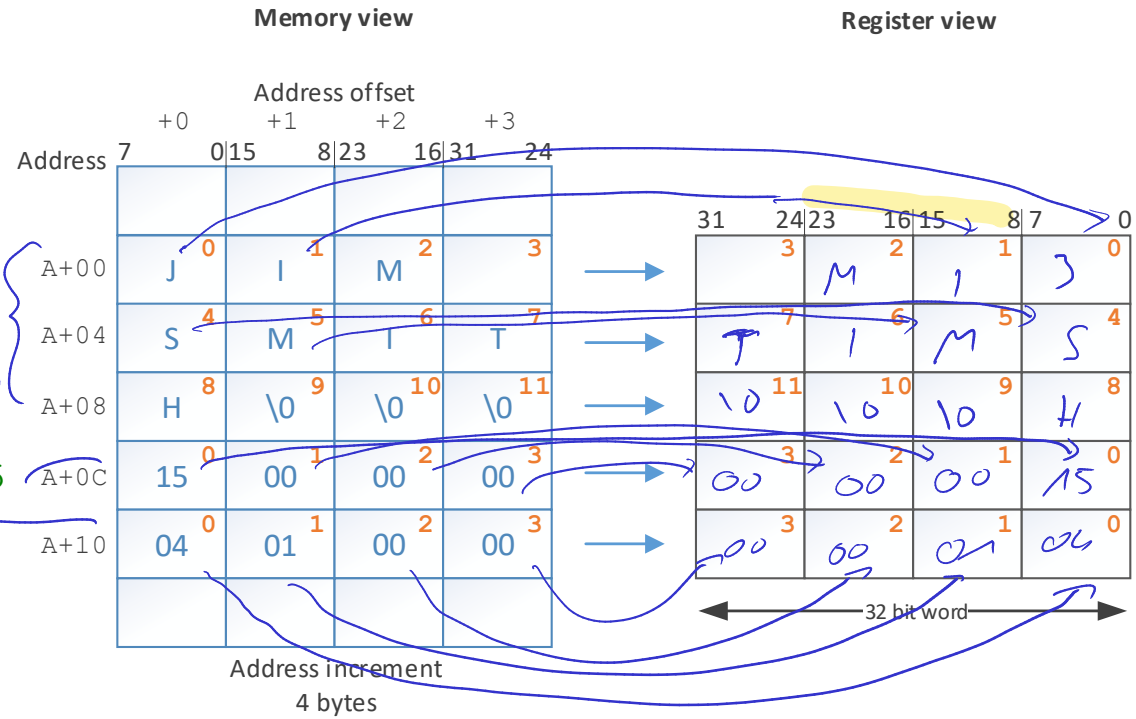
CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Endianness - example LE

Little endian memory -> Register

```
1  #include <stdlib.h>
2  #include <stdint.h>
3
4  int main()
5  {
6      struct employee {
7          char     name[12];
8          uint32_t age;
9          uint32_t dept_nr;
10     };
11
12     struct employee smith = {
13         .name    = "JIM SMITH",
14         .age     = 21,    //0x15
15         .dept_nr = 0x104 //260
16     };
17
18     return EXIT_SUCCESS;
19 }
```



[cmp: [1, p. 95-96]]

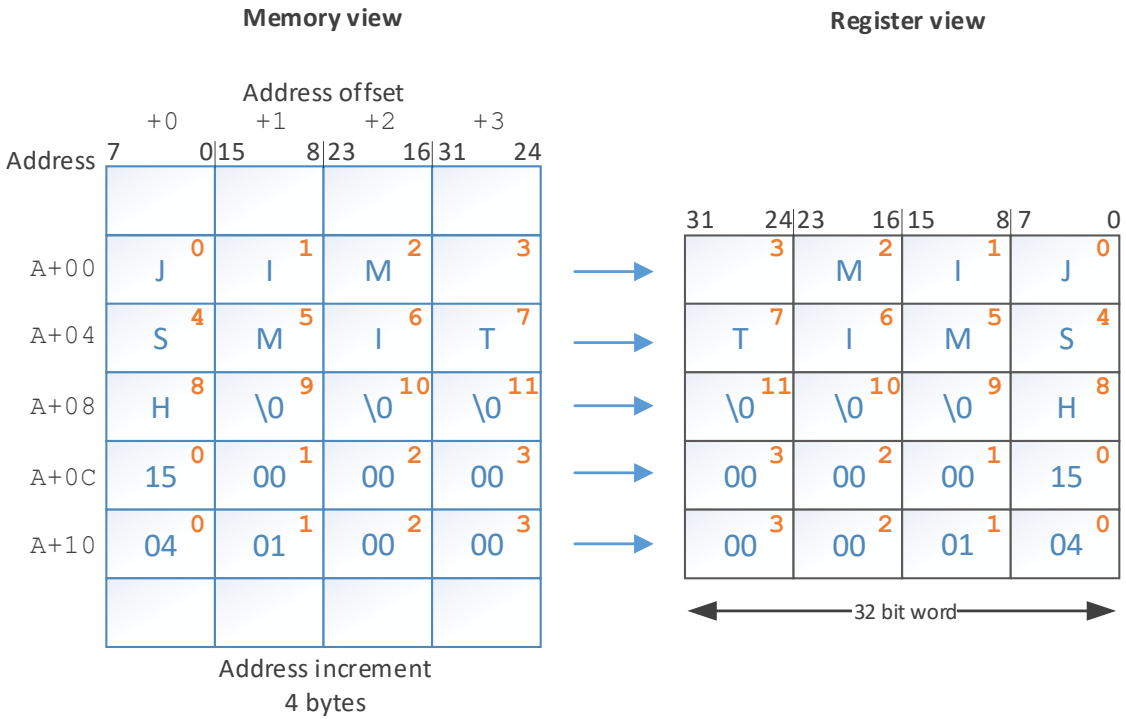
CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Endianness - example LE

Little endian memory -> Register

```
1  #include <stdlib.h>
2  #include <stdint.h>
3
4  int main()
5  {
6      struct employee {
7          char     name[12];
8          uint32_t age;
9          uint32_t dept_nr;
10     };
11
12     struct employee smith = {
13         .name    = "JIM SMITH",
14         .age     = 21,    //0x15
15         .dept_nr = 0x104 //260
16     };
17
18     return EXIT_SUCCESS;
19 }
```



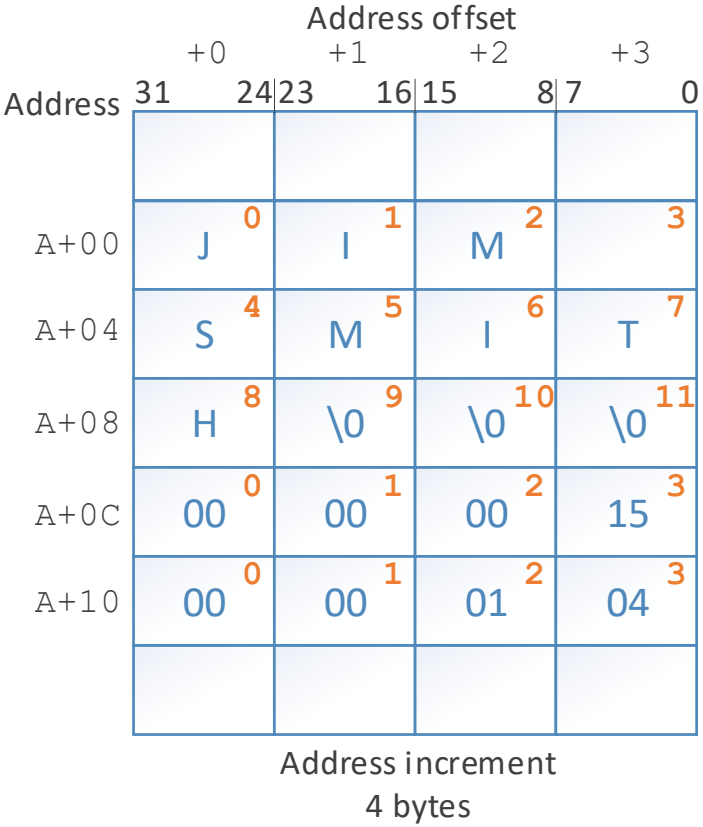
[cmp: [1, p. 95-96]]

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Endianness - example BE/LE

Big endian memory



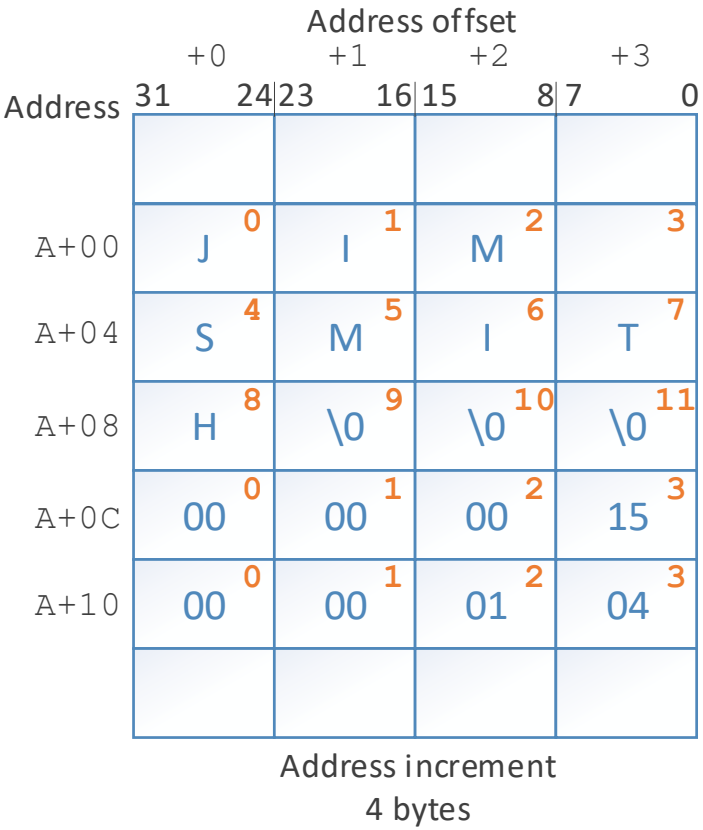
[cmp: [1, p. 95-96]]

CAMPUS Rosenheim
Computer Science

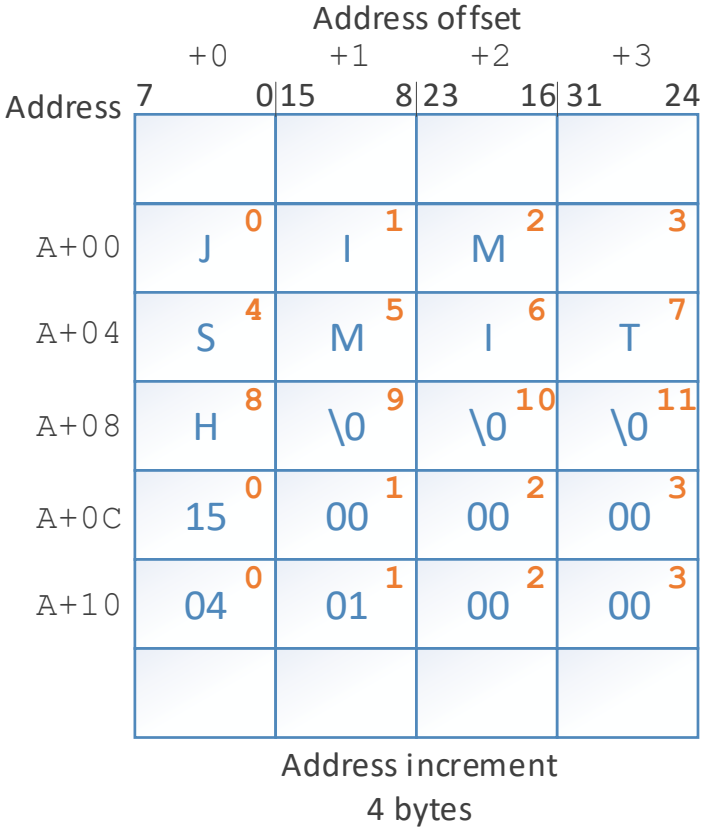
Technische Hochschule Rosenheim
Technical University of Applied Sciences

Endianness - example BE/LE

Big endian memory



Little endian memory



[cmp: [1, p. 95-96]]

Endianness - usage

Big endian

- IBM Mainframe
- Freescale ColdFire
- Atmel AVR/AVR32
- ARM Thumb and ARM64 (also Apple M1)

Little endian

- Intel x86
- x86-64 (AMD64, Intel 64)
- RISC-V
- Qualcomm Hexagon

Endianness - usage

Big endian

- IBM Mainframe
- Freescale ColdFire
- Atmel AVR/AVR32
- ARM Thumb and ARM64 (also Apple M1)

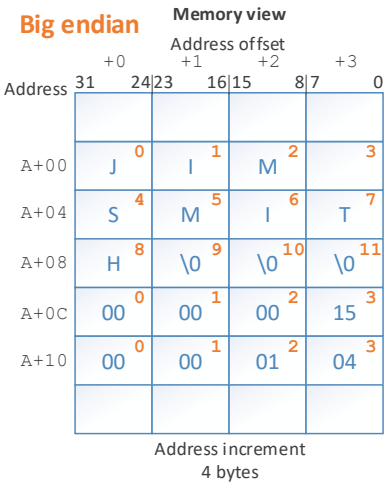
Little endian

- Intel x86
- x86-64 (AMD64, Intel 64)
- RISC-V
- Qualcomm Hexagon

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Endianness - transfer: BE to LE

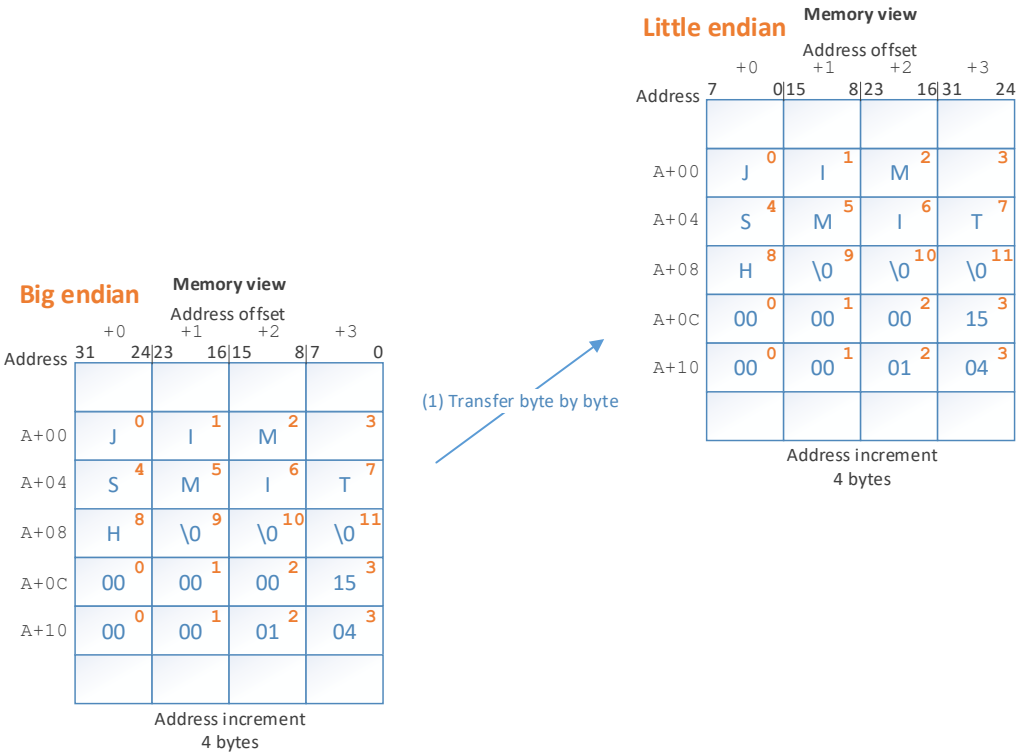


[cmp: [1, p. 95-96]]

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Endianness - transfer: BE to LE

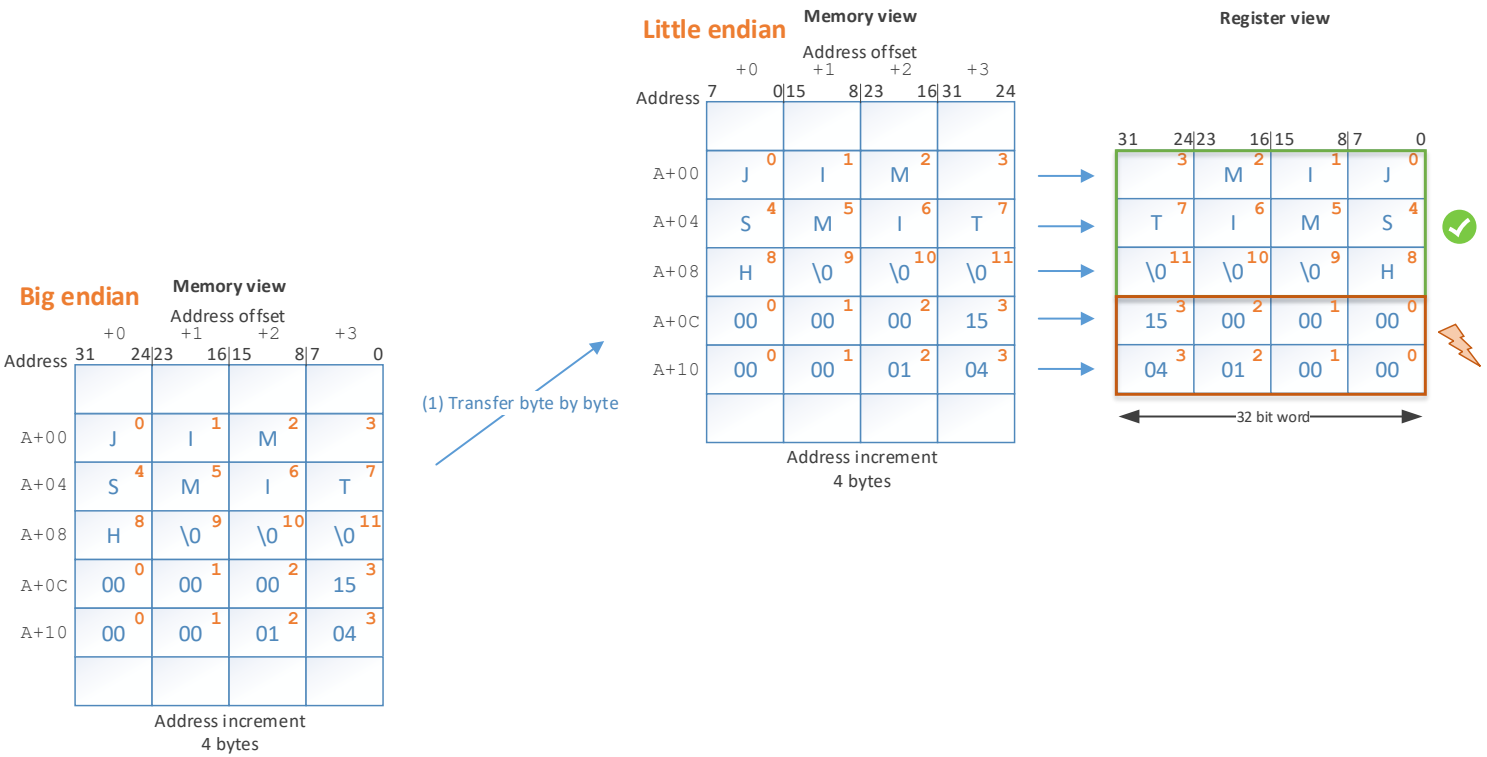


[cmp: [1, p. 95-96]]

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Endianness - transfer: BE to LE

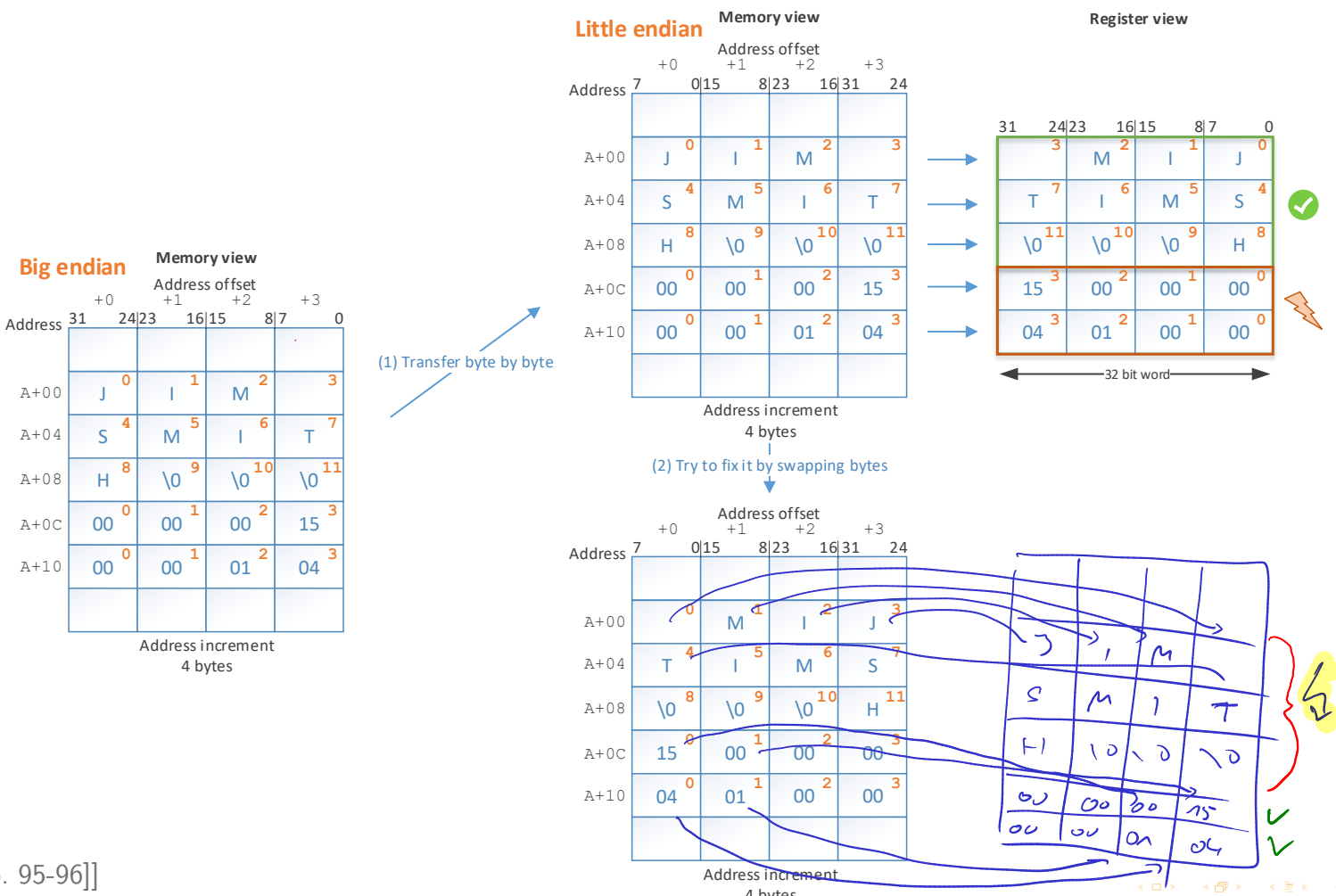


[cmp: [1, p. 95-96]]

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Endianness - transfer: BE to LE

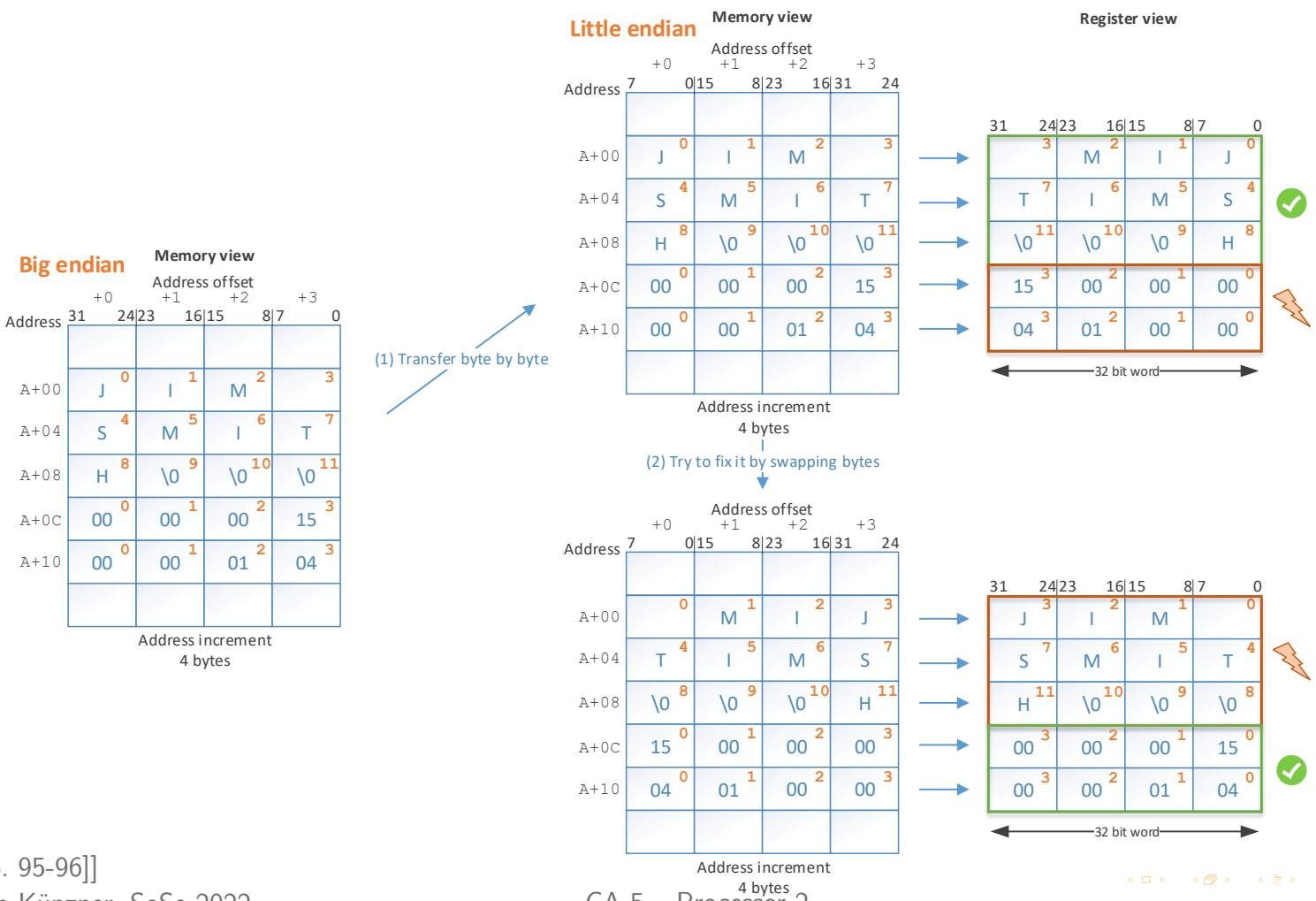


[cmp: [1, p. 95-96]]

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Endianness - transfer: BE to LE



[cmp: [1, p. 95-96]]

Endianness - problem

Problem can occur if

- Different data types are mixed: numbers, strings, or other data types
- Data type consists of more than one byte (multi-byte word, ≥ 2)
- Data are transferred between BE/LE systems

No problem occurs if

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Endianness - problem

Problem can occur if

- Different data types are mixed: numbers, strings, or other data types
- Data type consists of more than one byte (multi-byte word, ≥ 2)
- Data are transferred between BE/LE systems

No problem occurs if

Endianness - problem

Problem can occur if

- Different data types are mixed: numbers, strings, or other data types
- Data type consists of more than one byte (multi-byte word, ≥ 2)
- Data are transferred between BE/LE systems

No problem occurs if

Endianness - problem

Problem can occur if

- Different data types are mixed: numbers, strings, or other data types
- Data type consists of more than one byte (multi-byte word, ≥ 2)
- Data are transferred between BE/LE systems

No problem occurs if

Endianness - problem

Problem can occur if

- Different data types are mixed: numbers, strings, or other data types
- Data type consists of more than one byte (multi-byte word, ≥ 2)
- Data are transferred between BE/LE systems

No problem occurs if

- Single-byte data is transferred byte by byte (e.g. ASCII)
- Data is transferred within same endianness (LE -> LE, BE -> BE)

Endianness - problem

Problem can occur if

- Different data types are mixed: numbers, strings, or other data types
- Data type consists of more than one byte (multi-byte word, ≥ 2)
- Data are transferred between BE/LE systems

No problem occurs if

- Single-byte data is transferred byte by byte (e.g. ASCII)
- Data is transferred within same endianness (LE -> LE, BE -> BE)

Endianness - problem

Problem can occur if

- Different data types are mixed: numbers, strings, or other data types
- Data type consists of more than one byte (multi-byte word, ≥ 2)
- Data are transferred between BE/LE systems

No problem occurs if

- Single-byte data is transferred byte by byte (e.g. ASCII)
- Data is transferred within same endianness (LE -> LE, BE -> BE)

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Endianness - conclusion

Without the knowledge about the data types and the endianness, a transfer between BE/LE systems is not feasible.

Tanenbaum: „*There is no easy solution to this*" [1, p. 96]

Endianness - conclusion

Without the knowledge about the data types and the endianness, a transfer between BE/LE systems is not feasible.

Tanenbaum: „*There is no easy solution to this*“ [1, p. 96]

Endianness - possible solutions

Possible solution

- **Know** the endianness (e.g. **meta data!**)
- **Transfer** byte by byte (no problem for single-byte data)
- If endianness is different and a multi-byte word is transferred: additionally **swap** the bytes

Endianness - possible solutions

Possible solution

- **Know** the endianness (e.g. **meta data!**)
- **Transfer** byte by byte (no problem for single-byte data)
- If endianness is different and a multi-byte word is transferred: additionally **swap** the bytes

Endianness - possible solutions

Possible solution

- **Know** the endianness (e.g. **meta data!**)
- **Transfer** byte by byte (no problem for single-byte data)
- If endianness is different and a multi-byte word is transferred: additionally **swap** the bytes

Endianness - possible solutions

Possible solution

- **Know** the endianness (e.g. **meta data!**)
- **Transfer** byte by byte (no problem for single-byte data)
- If endianness is different and a multi-byte word is transferred: additionally **swap** the bytes

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Endianness - solutions

Some examples:

- Network order: always BE
- Java: always BE; for transfer with others, `ByteOrder` can be set
- Unicode UTF-16/32: uses a BOM (byte order mark)
- TIF files: BE/LE identifier in header
- RPC (remote procedure call): marshalling (data as byte stream) solves the problem by using meta data

Endianness - solutions

Some examples:

- Network order: always BE
- Java: always BE; for transfer with others, `ByteOrder` can be set
- Unicode UTF-16/32: uses a BOM (byte order mark)
- TIF files: BE/LE identifier in header
- RPC (remote procedure call): marshalling (data as byte stream) solves the problem by using meta data

Endianness - solutions

Some examples:

- Network order: always BE
- Java: always BE; for transfer with others, `ByteOrder` can be set
- Unicode UTF-16/32: uses a BOM (byte order mark)
- TIF files: BE/LE identifier in header
- RPC (remote procedure call): marshalling (data as byte stream) solves the problem by using meta data

Endianness - solutions

Some examples:

- Network order: always BE
- Java: always BE; for transfer with others, `ByteOrder` can be set
- Unicode UTF-16/32: uses a BOM (byte order mark)
- TIF files: BE/LE identifier in header
- RPC (remote procedure call): marshalling (data as byte stream) solves the problem by using meta data

Endianness - solutions

Some examples:

- Network order: always BE
- Java: always BE; for transfer with others, `ByteOrder` can be set
- Unicode UTF-16/32: uses a BOM (byte order mark)
- TIF files: BE/LE identifier in header
- RPC (remote procedure call): marshalling (data as byte stream) solves the problem by using meta data

Endianness - solutions

Some examples:

- Network order: always BE
- Java: always BE; for transfer with others, `ByteOrder` can be set
- Unicode UTF-16/32: uses a BOM (byte order mark)
- TIF files: BE/LE identifier in header
- RPC (remote procedure call): marshalling (data as byte stream) solves the problem by using meta data

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Summary and outlook

Summary

- Endianness
- Examples
- Usage
- Transfer
- Solutions

Outlook

- Processor registers
- Processor examples
- Addressing modes

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Summary and outlook

Summary

- Endianness
- Examples
- Usage
- Transfer
- Solutions

Outlook

- Processor registers
- Processor examples
- Addressing modes