School of Engineering
InES Institute of Embedded Systems

Remaining Data Types

CT Team: A. Gieriet, J. Gruber, R. Gübeli, M. Meli, M. Rosenthal, A. Rüst, J. Scheier, M. Thaler

Contents

- **Real numbers**
  - Single vs. double precision
- **Arrays**
  - One dimensional
  - Pointer and Reference
  - Multi dimensional
- **Strings**
- **Structs**

Learning Objectives

- At the end of this lesson you will be able
  - to explain the structure and notation of real numbers in decimal and binary
  - to translate real numbers between decimal and binary
  - to explain how arrays are stored and how to access to single elements
  - to discuss how structs and strings are stored

Scalar Types: Real Numbers

- **Real numbers: general notation**

$$\text{Value} = \text{Sign} \cdot \text{Fraction} \cdot \text{Base}^{\text{Exponent}}$$

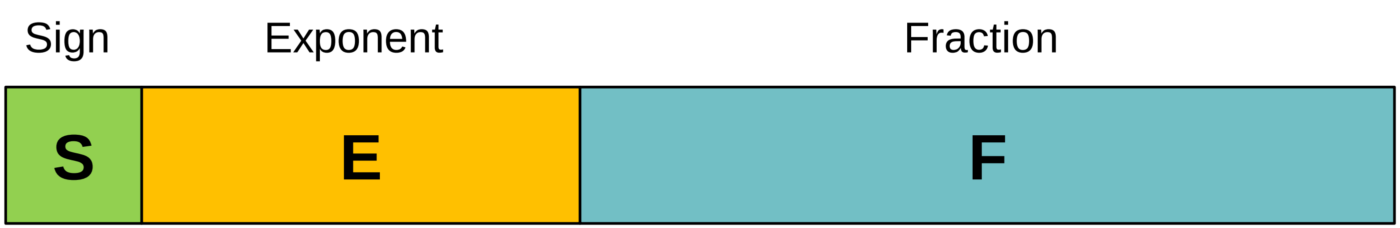
- **Normalized scientific notation**
  - Single non-zero digit to the left of the decimal (binary) point

- **Decimal vs. binary**
  - Decimal: 0.00002796 $= 2.796 \cdot 10^{-5}$
  - Binary: 0.000010110111 $= 1.0110111 \cdot 2^{-5}$

IEEE Standard 754 / 854

- **Defines how floating point numbers are represented**
  - Easy exchange of data between machines
  - Simplifies hardware algorithms

- **Binary notation**



IEEE Standard 754 / 854

Fraction

- Binary notation
- Representing normalized numbers → number of form ***1***.xxxx..
- In IEEE 754 standard, the ***1*** is implicit

$$\boxed{\text{Fraction value} = (1 + F)}$$

- Example

$$\boxed{1.265625\ d = 1.010001\ b \rightarrow F = 010001}$$

Remark: 1.010001 b = $(1 + 0 \times 2^{-1} + 1 \times 2^{-2} + \ldots + 1 \times 2^{-6})$ d

IEEE Standard 754 / 854

Exponent

- Binary excess notation

  Exponent value = (E - Bias)

- Bias: 127 (float) / 1023 (double)

- Example (float):

  $0111'1010b = 122 \rightarrow 122 - 127 = -5 \rightarrow \text{Value} = 2^{-5}$

  $\text{Value} = 2^{-5} \rightarrow -5 + 127 = 122 = 0111'1010b$

IEEE Standard 754 / 854

- **Sign**

  - 1 bit
  - Sign value = $(-1)^S$

IEEE Standard 754 / 854

Sign and Magnitude Representation

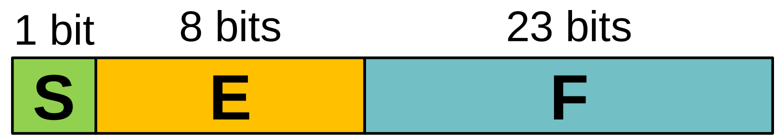
S	E	F

$$\text{Value} = (-1)^{S} \cdot (1 + F) \cdot 2^{(E\,-\,\text{Bias})}$$

- More exponent bits → wider range of numbers
- More fraction bits → higher precision

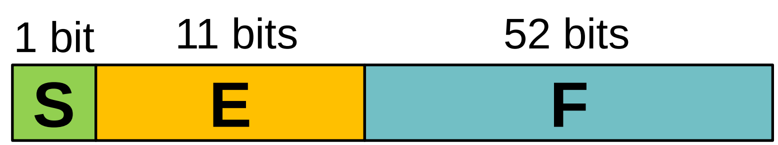
IEEE Standard 754 / 854

- **Single precision (float)**



  - 32 bits
  - Exponent bias: 127
  - Dynamics: $-3.4 \cdot 10^{38}$ ... $-1.17 \cdot 10^{-38}$, 0, $1.17 \cdot 10^{-38}$ ... $3.4 \cdot 10^{38}$
  - Resolution: $2^{-24} = 0.6 \cdot 10^{-7} \rightarrow$ 7 digits

- **Double precision (double)**



  - 64 bits
  - Exponent bias: 1023
  - Dynamics: $-1.8 \cdot 10^{308}$ ... $-2.2 \cdot 10^{-308}$, 0, $2.2 \cdot 10^{-308}$ ... $1.8 \cdot 10^{308}$
  - Resolution: $2^{-53} = 0.11 \cdot 10^{-15} \rightarrow$ 15 digits

Scalar Types: Single Precision

- **Special Values**

E	F	S	Result
255	$\neq 0$		not a number
255	0	1	$- \infty$
255	0	0	∞
$0 < E < 255$			$(-1)^S \cdot (1 + F) \cdot 2^{(E\text{-}Bias)}$
0	$\neq 0$		$(0+F) \cdot 2^{(-Bias+1)}$
0	0	1	$- \ 0$
0	0	0	$+ 0$

Exercise: Single Precision (float)

- **Decimal 7.5 d = ?**

Exercise: Single Precision (float)

- **Binary 10111111'01010000'00000000'00000000 = ?**

Arrays in C

- **Pointer to first element of array**

```
int main(void) {
    int j, sum = 0;
    int array[5];
    int *ptr;                              // pointer to integer
    for (j = 0; j < 5; j++) {
        array[j] = j;
    }
    ptr = array;                           // copy pointer
    for (j = 0; j < 5; j++) {
        sum = sum + *ptr;                  // add value ptr points
        ptr++;                             // point to next element
    }
}
```

```
for (j = 0; j < 5; j++) {
    sum = sum + array[j];
}
```

Arrays in C

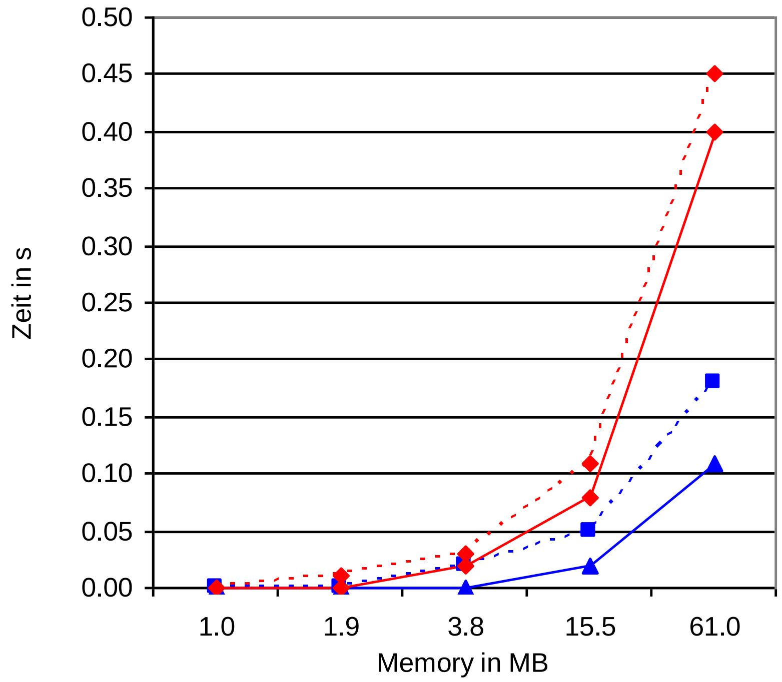
Multidimensional arrays are "Arrays of Arrays"

```
void main(void) {
   int array [3][3];
   int *ptr;
   int i, j, k;
   j = 0;                              // fill array with ascending
   for (i = 0; i < 3; i++) {           // numbers
      for (k = 0; k < 3; k++) {
         array [i][k] = j++;
   }  }
   ptr = array [0];                    // get pointer to first element
   for (i = 0; i < 9; i++) {
      printf("index: %d, value: %d\n", i, *ptr);
      ptr++;
   }  }
```

- Array a[3][3]: $\begin{bmatrix} a00 & a01 & a02 \\ a10 & a11 & a12 \\ a20 & a21 & a22 \end{bmatrix}$
- Is stored as: $\begin{bmatrix} a00 & a01 & a02 & a10 & a11 & a12 & a20 & a21 & a22 \end{bmatrix}$

Arrays: Who Cares?

- **Access to arrays: g++, Linux**
  - long int array [N][N]
  - N = 500, 700, 1000, 2000, 4000, 8000, 12000, 16000, 20000
    M = 1, 1.9, 3.8, 15.3, 61.0, 244, 550, 980, 1500 MB



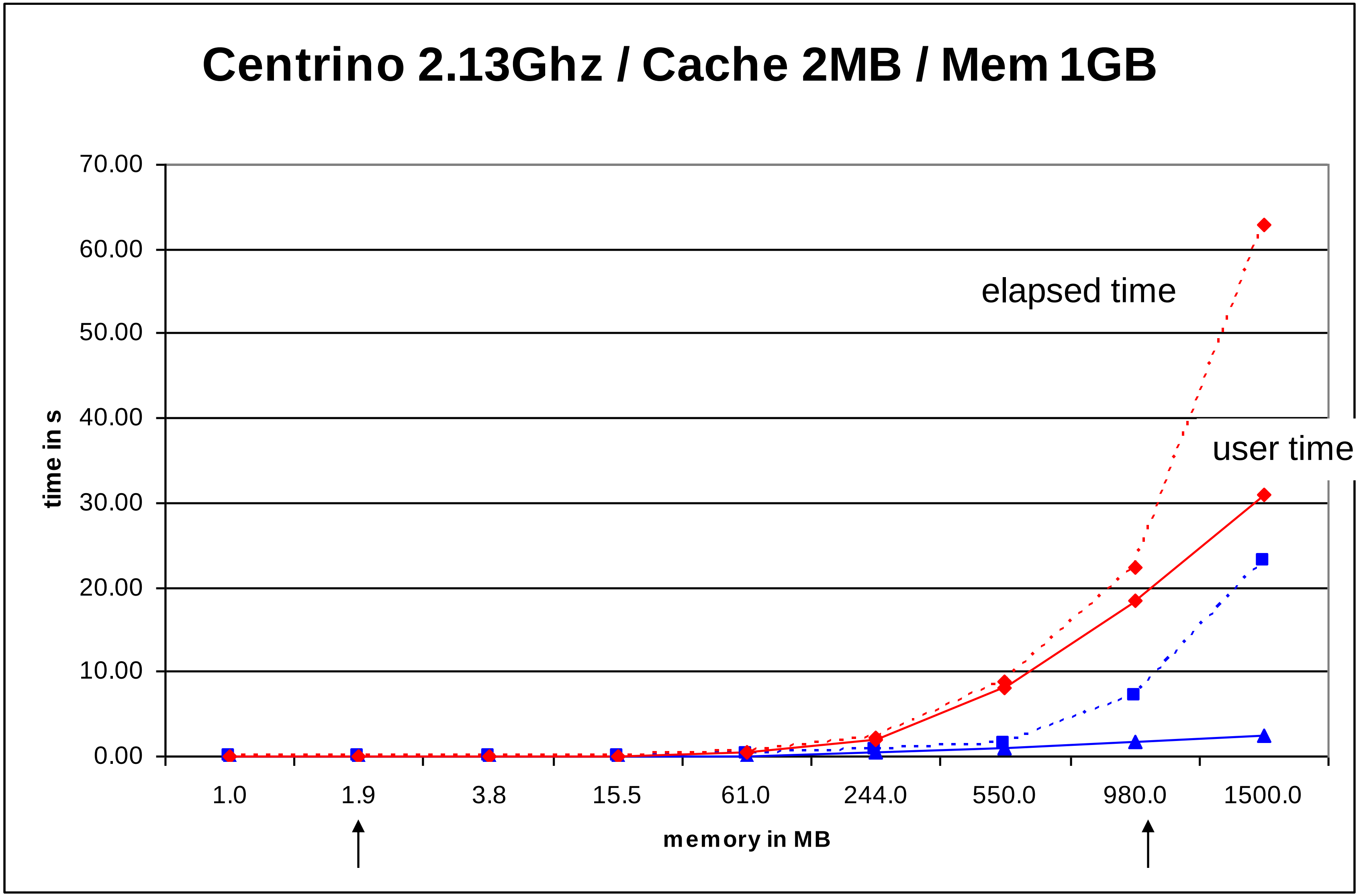
- Blue: CPU-time

```
for (i = 0; i < s; i++) {
    for (j = 0; j < s; j++) {
        array[i][j]=i+j;
}
```

- Red: CPU-time

```
for (i = 0; i < s; i++) {
    for (j = 0; j < s; j++) {
        array[j][i]=i+j;
}
```

... arrays



Centrino 2.13Ghz / Cache 2MB / Mem 1GB
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.00
time in s
elapsed time
user time
1.0
1.9
3.8
15.5
61.0
244.0
550.0
980.0
1500.0
memory in MB

Animation

Strings in C

School of Engineering
InES Institute of Embedded Systems

Strings in C are not objects

- Declaration with char arrays
- Example

```
char Name[16] = "Meier";
```

- Question: How is the end of a string recognised?
- All Strings in C are 0 – terminated
  → after the last character follows a '0'

'M'	'e'	'i'	'e'	'r'	'\0'	'i'	'e'	'a'	'Q'	'u'	'a'	'r'	'\0'	'l'	'l'

valid String

Terminator

undefined content

- Length of strings not stored → count characters to '\0'

Struct in C

- **Structs contain different data that belong together**
  - Declaration

```
struct person {
  char      name[8];
  char      prename[8];
  int       pers_nr;
  unsigned  char day_of_birth;
  unsigned  char month_of_birth;
};
```

  - Definition

```
struct person aperson;
```

  - Usage

```
aperson.Name = "Meier";
aperson.prename = "Peter";
```

Struct Memory Layout

School of Engineering
InES Institute of Embedded Systems

Access to struct

```
struct person {
  char      name[8];
  char      prename[8];
  int       pers_nr;
  unsigned  char day_of_birth;
  unsigned  char month_of_birth;
};
```

- Using base address

- name: base address + 0
- prename: base address + 8
- pers_nr: base address + 16
- day_of_birth: base address + 18
- month_of_birth: base address + 19

Identifier	Memory	Address
month_of_birth	8	2013h
day_of_birth	15	2012h
	034	2011h
pers_nr	156	2010h
		200Fh
		200Eh
	'\0'	200Dh
	r	200Ch
	e	200Bh
	t	200Ah
	e	2009h
prename	P	2008h
		2007h
		2006h
	'\0'	2005h
	r	2004h
	e	2003h
	i	2002h
	e	2001h
name	M	2000h

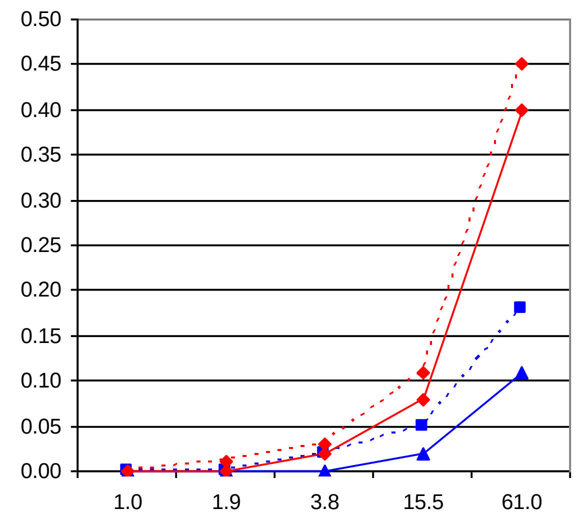
Base address

Conclusion

zh aw School of Engineering
InES Institute of Embedded Systems

- **Real numbers**
  - IEEE Standard 754 / 854
  - Single vs. double precision

1 bit	8 bits	23 bits
S	**E**	**F**



- **Memory Layout of**
  - Arrays
  - Strings
  - Structs

'M'	'e'	'i'	'e'	'r'	'\0'	'i'	'e'	'a'	'Q'	'u'	'a'	'r'	'\0'	'l'	'l'