REGIONALES HOCHSCHUL-
RECHENZENTRUM KAISERSLAUTERN

TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

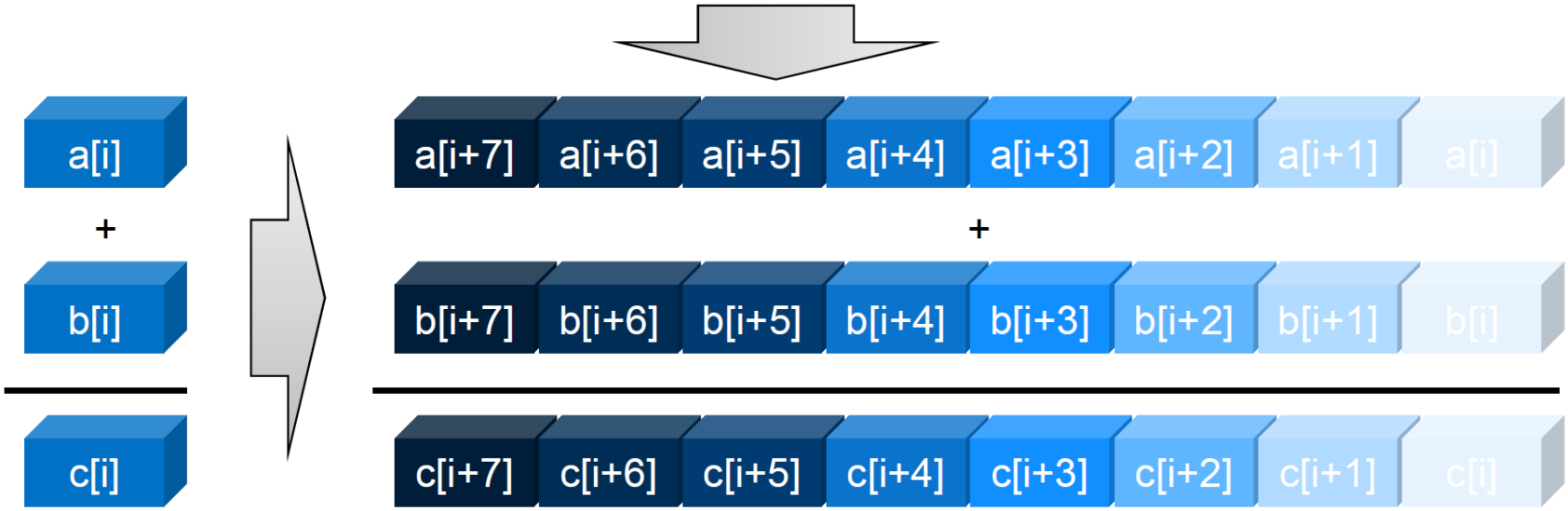
RHRK-Tutorial

Vectorization: Introduction

Course instructor: Gabriele Monaco
In charge: Dr. Josef Schüle, RHRK

What is vectorization

```
for(i = 0; i <= MAX;i++)
    c[i] = a[i] + b[i];
```



A single instruction execution on the left is transformed to a packed SIMD instruction operating on 8 doubles at a time on the right.

What is vectorization

- Loop unrolling
- Packed SIMD instructions
- The processor can execute instructions on multiple double at a time
- Initially supported on 128 bits registers (XMM0 to XMM15)
  - SSE extension (Streaming SIMD)
- Advanced vector extension with 256 bits registers (YMM0 to YMM15)
  - Further improved in AVX2
  - Available on Sandybridge (E5-2670), Haswell (E5-2640v3) and newer
- Latest architecture extension supporting 512 bits (ZMM0 to ZMM31)
  - AVX-512 supports many flavours (see later)
  - Backward compatible, lower bits are AVX and SSE extension registers
  - Available on Skylake (SP 6126) and newer

Why vectorization?

```
#define MAX 65536

int main(int argc, char *argv[]) {
    int i, j;
    float a[MAX], b[MAX], c[MAX];
    for (j=0;j<MAX;j++)
        for (i=0;i<MAX;i++)
            c[i]=a[i]+b[i];
    return c[MAX/2];
}
```

```
$ icc main.c -o main_novec
$ time ./main_novec

real    0m1.791s
user    0m1.782s
sys     0m0.004s
```

```
$ icc -xCOMMON-AVX512 main.c -o main_vec
$ time ./main_vec

real    0m0.305s
user    0m0.301s
sys     0m0.004s
```

Intel AVX extensions (present in elwe)

Name	Instruction set	#float	#doubles
empty	No AVX	/	/
avx	set 1 - limited	8	4
avx2	set 2 - advanced	8	4
avx512f	foundation	16	8
avx512cd	conflict detection	16	8
avx512dq	double word	16	8
avx512bw	byte and word	16	8
avx512vl	vector length	16	8

Verify running this in a terminal:

```
cat /proc/cpuinfo | grep avx | uniq | tr ' ' '\n' | grep avx
```

Intel AVX extensions (not in elwe)

Name	Instruction set	#float	#doubles
avx512er	exponential and reciprocal	16	8
avx512pf	prefetch	16	8
avx512ifma	integer fused multiply add	16	8
avx512vbmi	vector byte manipulation	16	8
avx5124fmaps	fused multiply accumulation single	16	8
avx5124vnniw	vector neural network word variable	16	8
avx512vpopcntdq	vector population count	16	8
avx512vnni	vector neural network	16	8
avx512vbmi2	vector byte manipulation 2	16	8
avx512bitalg	bit algorithms	16	8
avx512vp2intersect	vector pair intersection	16	8
avx512bf16	acceleration for bfloat16	16	8

A simple vectorized loop

```
//main.c

double a[MAX], b[MAX], c[MAX];

for(int i=0; i<MAX; i++)
    c[i] = a[i]+b[i];
```

```
//main.s

vmovapd    -32816(%rbp,%rax), %zmm1
vaddpd     -49200(%rbp,%rax), %zmm1, %zmm0
vmovapd    %zmm1, -49264(%rbp)
vmovapd    %zmm0, -16432(%rbp,%rax)
```

Compile with the -S flag to see the assembly code
Here special packed double precision (pd) instructions are used on the avx512 registers, the code is vectorized but the compiler should be instructed.

How to actually vectorize?

- Explicit inline assembly (usually not needed)
- Write normal loops in C code and let the compiler decide
  - Based on it's knowledge it may understand whether vectorization would bring an improvement or not
- Give additional information to the compiler
- Explicitly force the compiler to vectorize
- In general keep things simple
  - Easier said than done..
  - Keep following here!

- **High Performance Computing on Elwetritsch**

Vielen Dank
Thank You