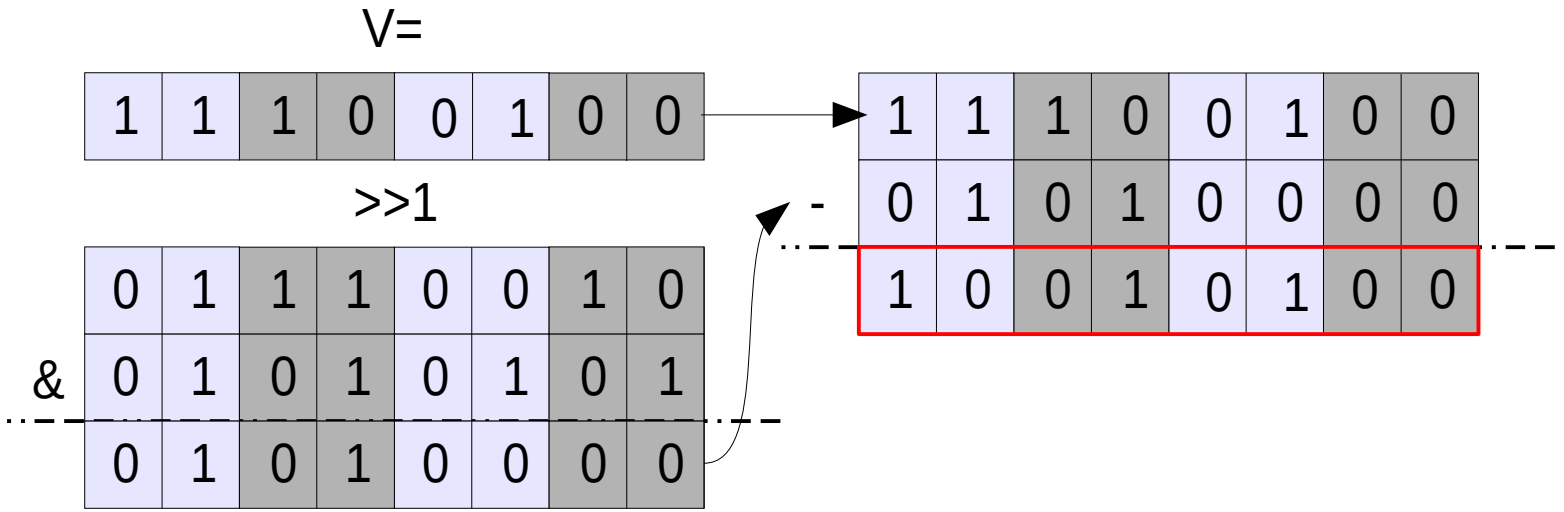
Count Bits Set

German Language
Aus meiner Vorlesung

Count Bits Set (in an Integer)

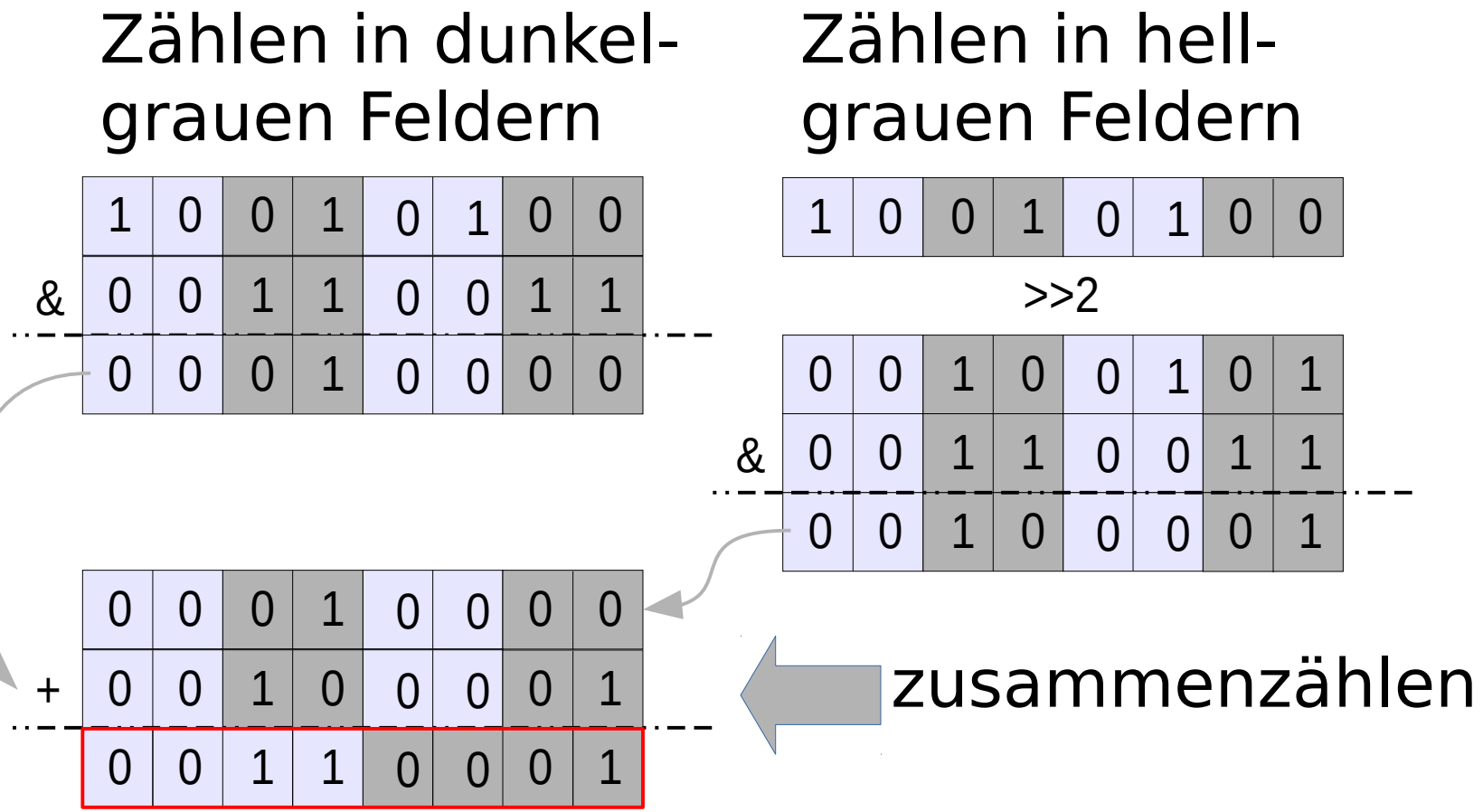
```
uint32_t cbs(uint32_t v){
  uint32_t c;
  // reuse input as temporary:
  v = v - ((v >> 1) & 0x55555555);                    //count in fields of size 2
  v = (v & 0x33333333) + ((v >> 2) & 0x33333333);     //add fields of size 2 => now fields of size 4
  c = ((v + (v >> 4) & 0xF0F0F0F) * 0x1010101) >> 24; //from size4 add results to fields of size 8,
                                                      //plus add all results of fields with size 8
                                                      //and shift result to lowest Byte
  return c;
}
```

Schritt 1: Zählen in Feldern der Größe 2



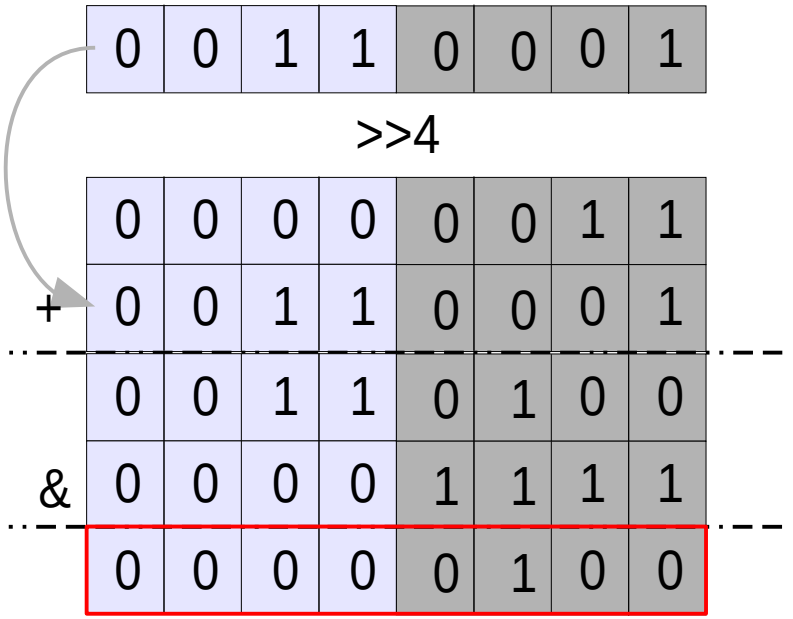
Count Bits Set (in an Integer)

Schritt 2: Felder der Größe 2 zusammenfassen zu Feldern der Größe 4



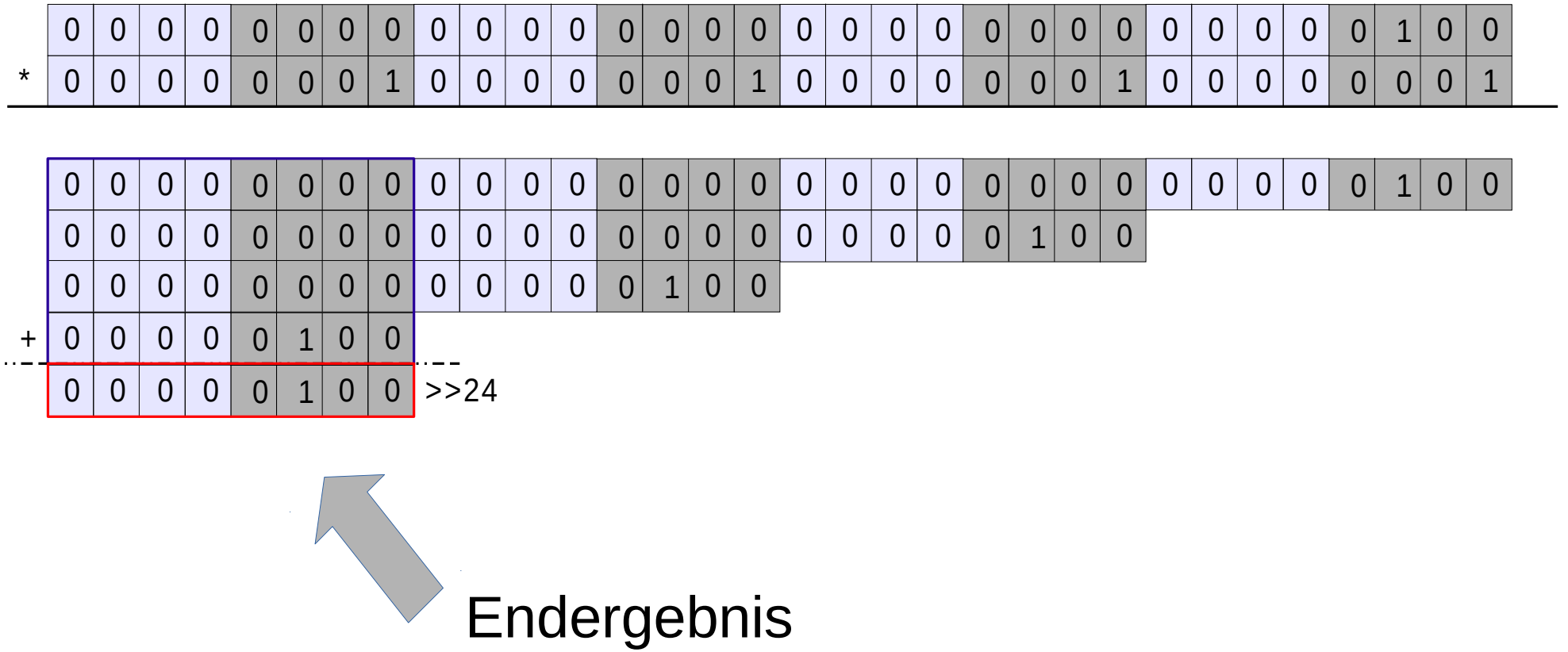
Count Bits Set (in an Integer)

Schritt 3: Felder der Größe 4 zusammenfassen
zu Feldern der Größe 8



Count Bits Set (in an Integer)

Schritt 4: 4 Felder der Größe 8 zusammenfassen zu Feld der Größe 32



Vergleich von Lösungen des Problems

X68 Assembler

maskierender Vergleich

```
17  uint32_t simple(uint32_t v){
18    uint32_t c = 0;
19    for(uint32_t ii=0x8000; ii>0; ii>>=1){
20      if((ii&v)>0){
21        ++c;
22      }
23    }
24    return c;
25  }
```

```
0x0040388C      mov     $0x10,%edx
0x00403891      xor     %ecx,%ecx
0x00403893      mov     $0x8000,%eax
0x00403898      nop
0x00403899      lea     0x0(%esi,%eiz,1),%esi
0x004038A0      mov     %eax,%ebx
0x004038A2      and     %esi,%ebx
0x004038A4      cmp     $0x1,%ebx
0x004038A7      sbb     $0xffffffff,%ecx
0x004038AA      shr     %eax
0x004038AC      sub     $0x1,%edx
0x004038AF      jne     0x4038a0 <_fu1___ZSt4cout+10
0x004038B1      mov     %ecx,(%esp)
0x004038B4      mov     $0x6fcc43c0,%ecx
```

X 32

„paralleles“ Zählen

```
6   uint32_t cbs(uint32_t v){
7     uint32_t c;
8     // reuse input as temporary:
9     v = v - ((v >> 1) & 0x55555555);
10    v = (v & 0x33333333) + ((v >> 2) & 0x33333333);
11    c = (((v + (v >> 4)) & 0xF0F0F0F) * 0x1010101) >> 24;
12
13
14    return c;
15  }
```

```
0x00403811      mov     %eax,%edx
0x00403813      shr     %edx
0x00403815      and     $0x55555555,%edx
0x0040381B      sub     %edx,%eax
0x0040381D      mov     %eax,%edx
0x0040381F      shr     $0x2,%eax
0x00403822      and     $0x33333333,%edx
0x00403828      and     $0x33333333,%eax
0x0040382D      add     %edx,%eax
0x0040382F      mov     %eax,%edx
0x00403831      shr     $0x4,%edx
0x00403834      add     %edx,%eax
0x00403836      and     $0xf0f0f0f,%eax
0x0040383B      imul    $0x1010101,%eax,%eax
0x00403841      shr     $0x18,%eax
0x00403844      mov     %eax,(%esp)
```

Vergleich von Lösungen des Problems

```
Benchmarking Algorithms for "Count Set Bits"
Running all Algorithms with Input from 0 to 4294967295...
Algorithmus 0:   69  Parallel                -> Version of CountBitsSet
Algorithmus 1:   71  Parallel 2              -> Version of CountBitsSet
Algorithmus 2:  154  Nibble LUT              -> Nibble only
Algorithmus 3:  140  Nibble LUT2             -> 2 Nibbles of a Byte at once
Algorithmus 4:   93  Nubble LUT Unrolled     -> Access Bytes by shifts
Algorithmus 5:  110  Nubble LUT Unrolled2    -> Access Bytes via union
Algorithmus 6:  291  briankernighansway      -> Human readable Algorithm
```

Dauer in Sekunden