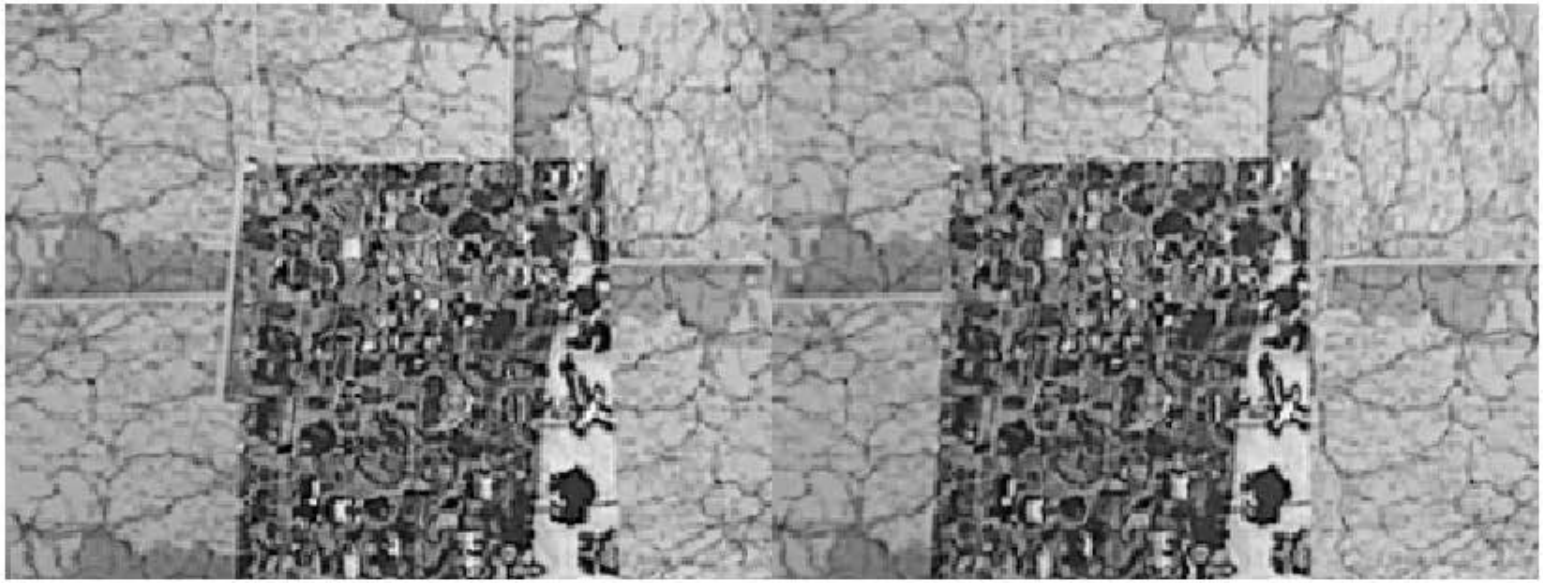
Computer System

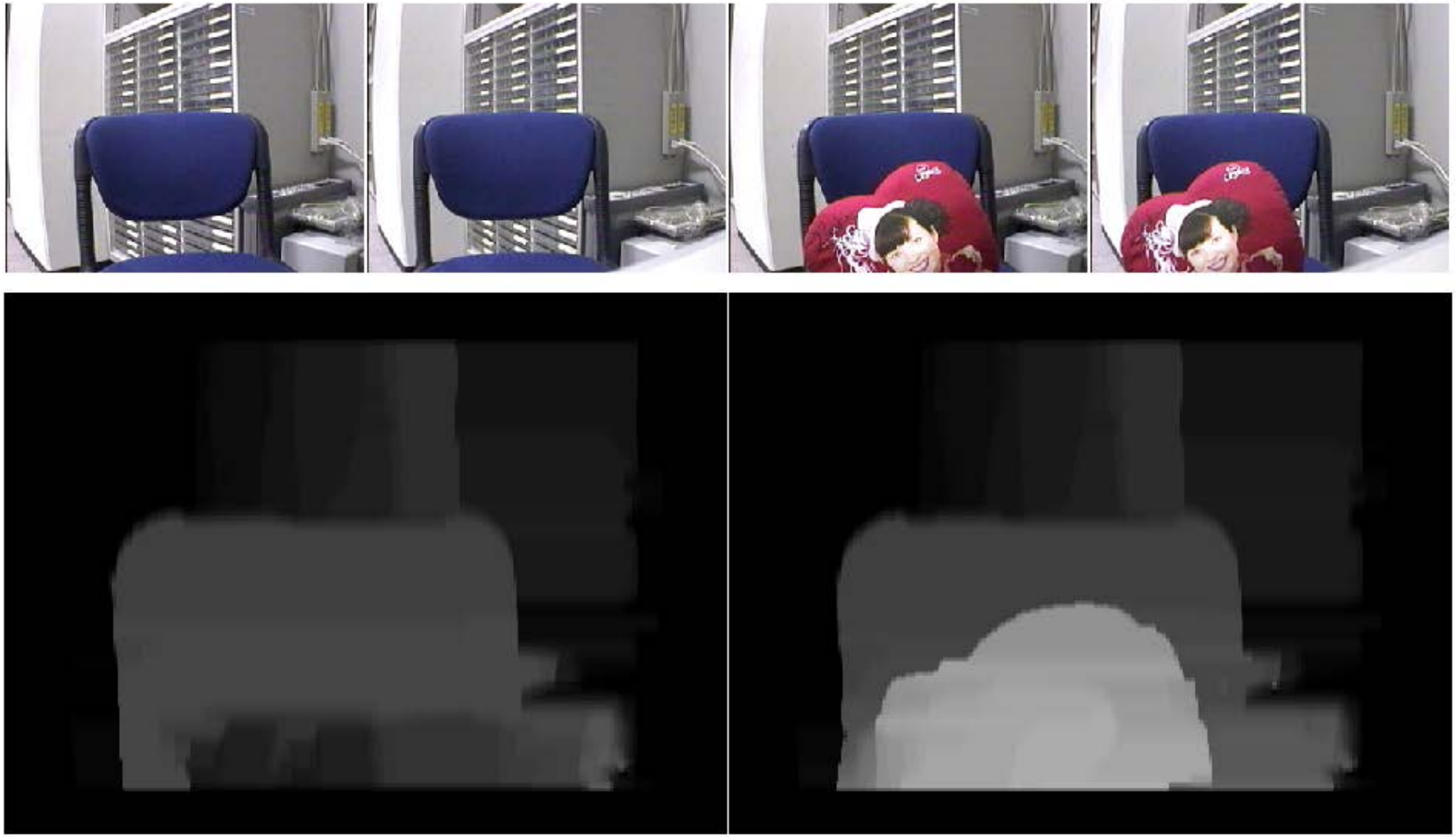
Step2. CA0302:High performance media processing and VLIW

http://archlab.naist.jp/Lectures/ARCH/ca0302/ca0302e.pdf

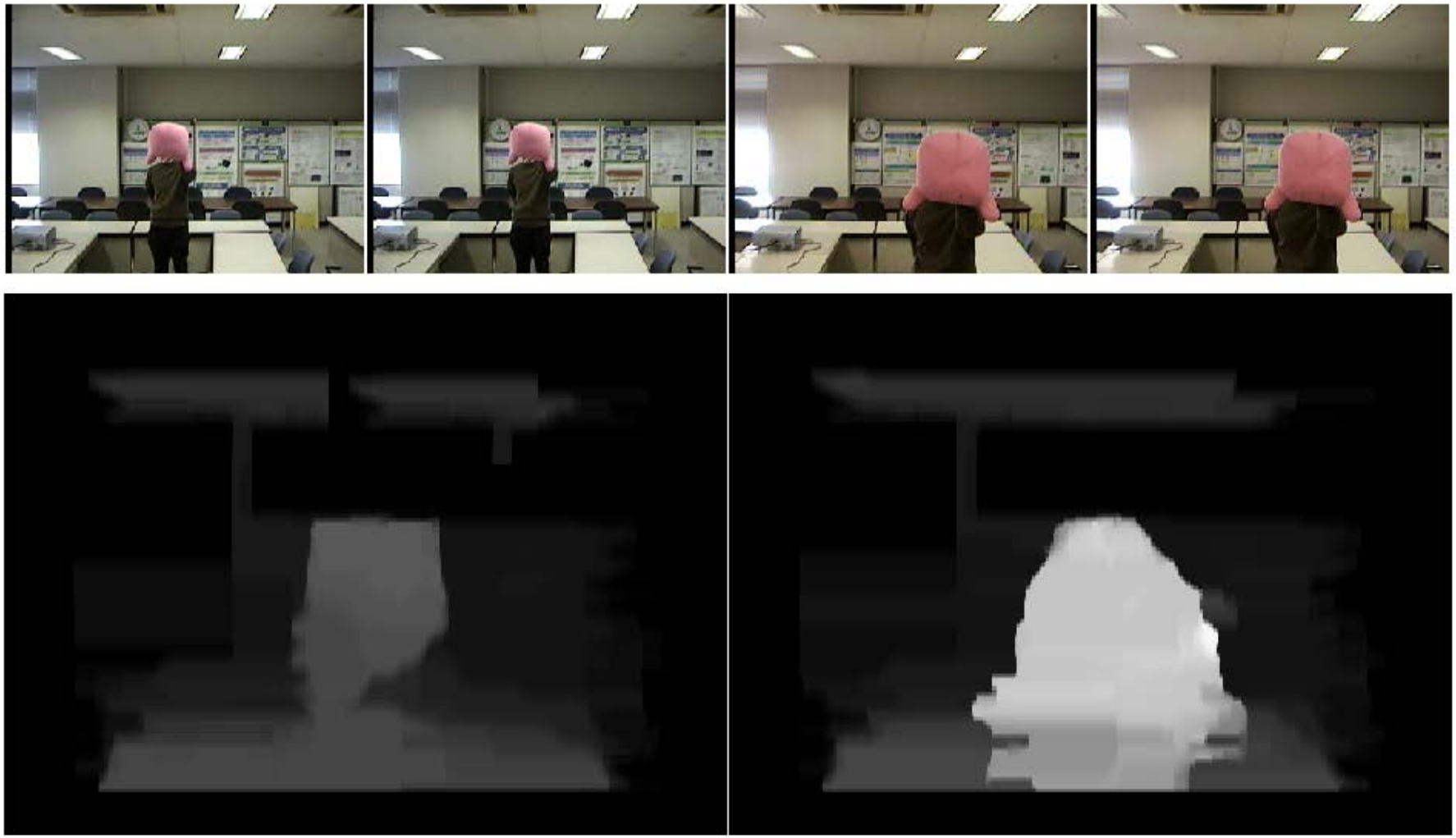
Stereo matching



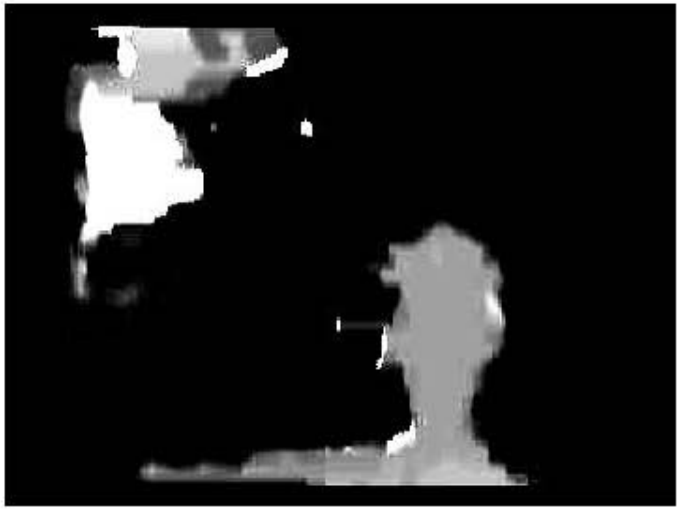
Object detection



Object detection

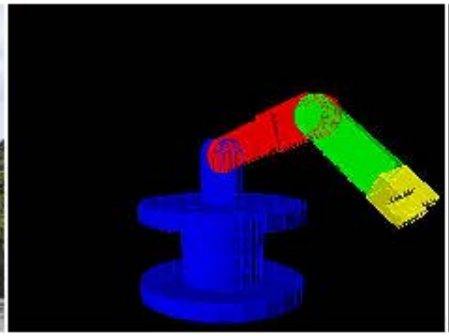
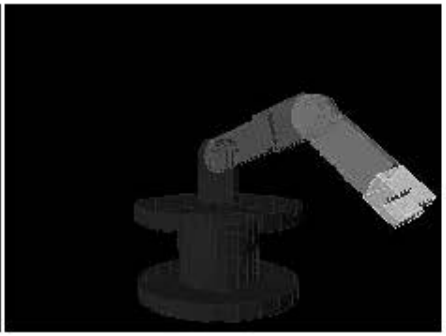
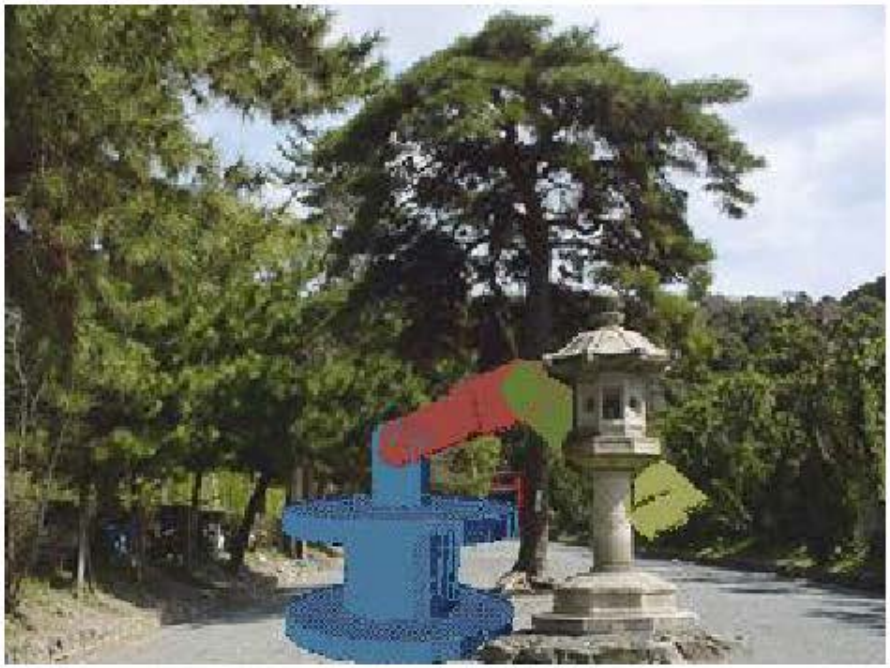


Object extraction

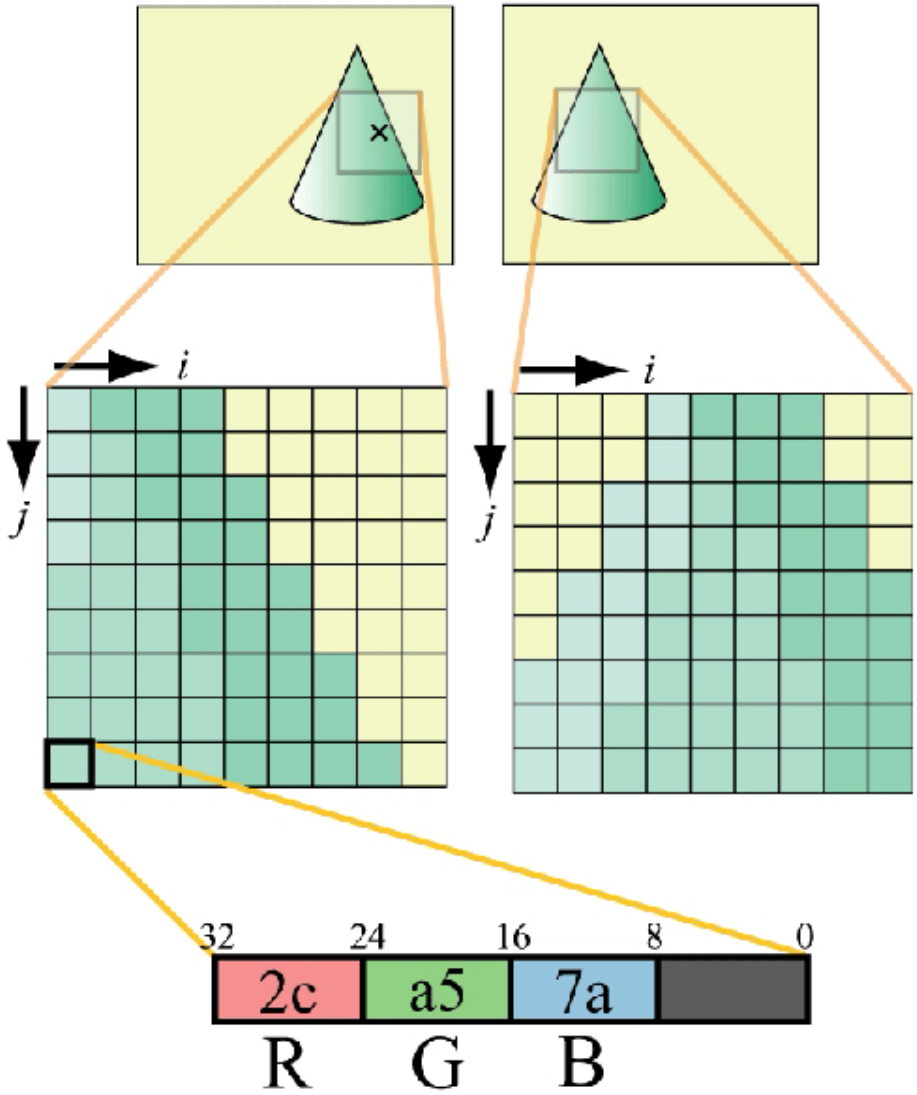


Z-keying



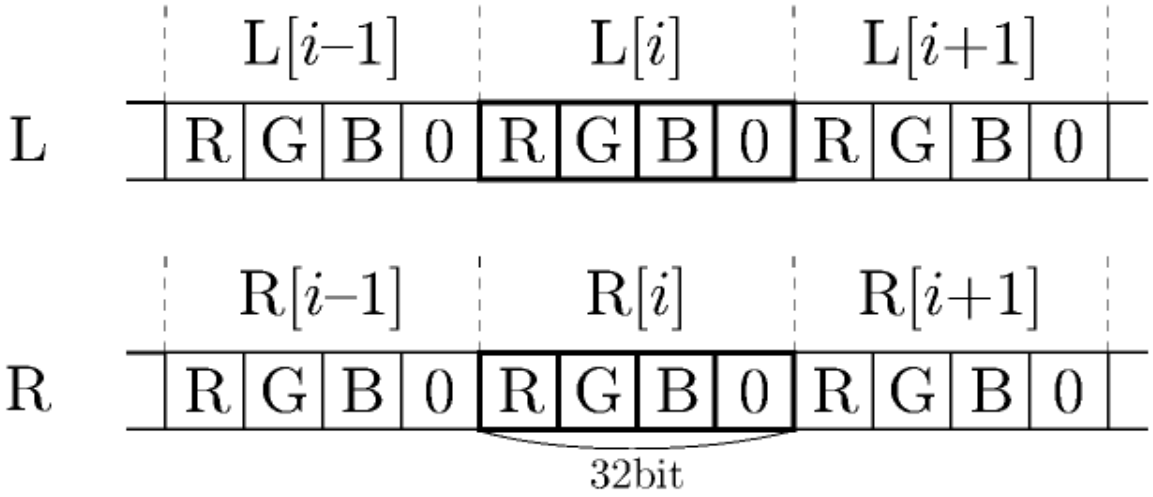
Stereo matching requires heavy computation



How to find similar image

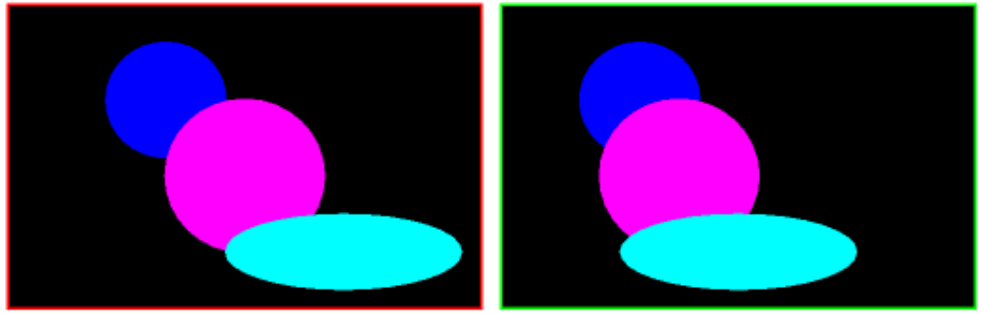
SAD (Sum of Absolute Difference)

- **Sliding 21x21 pixel window**
- **Each pixel has 4bytes (RGB0)**



$$DIFF_i = |L[i]_R - R[i]_R| + |L[i]_G - R[i]_G| + |L[i]_B - R[i]_B|$$

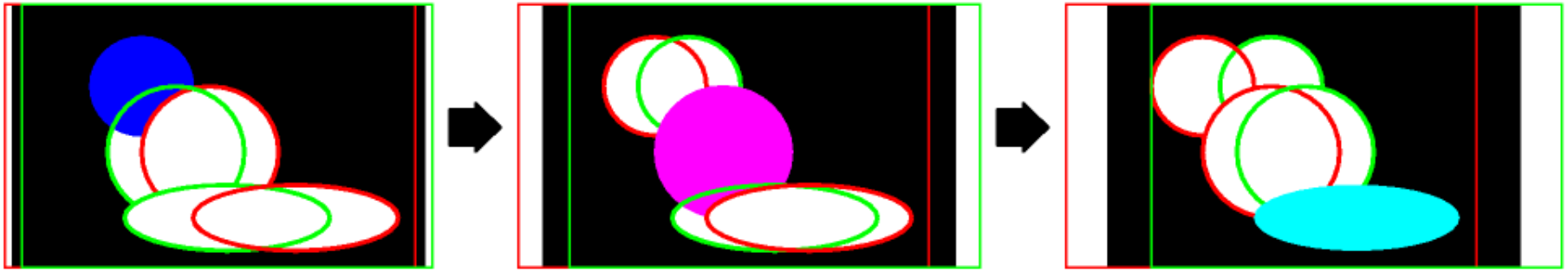
Stereo matching based on edges

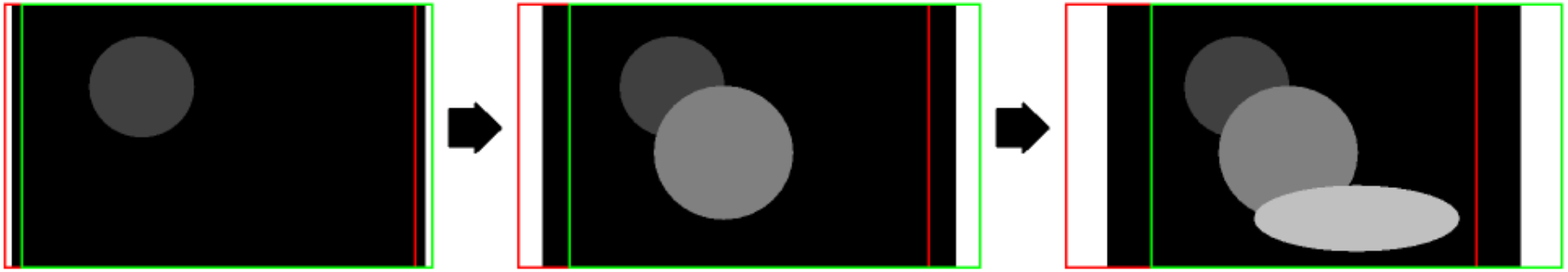


Focus on far side

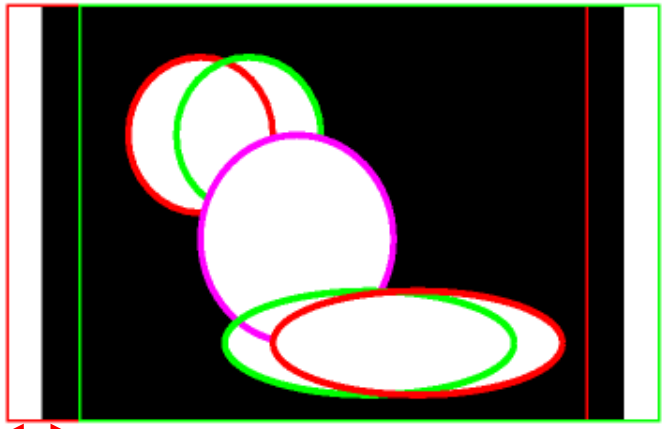
Mid

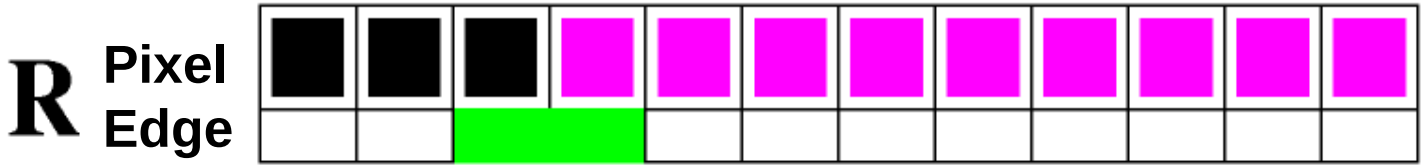
Near





Stereo matching based on edges

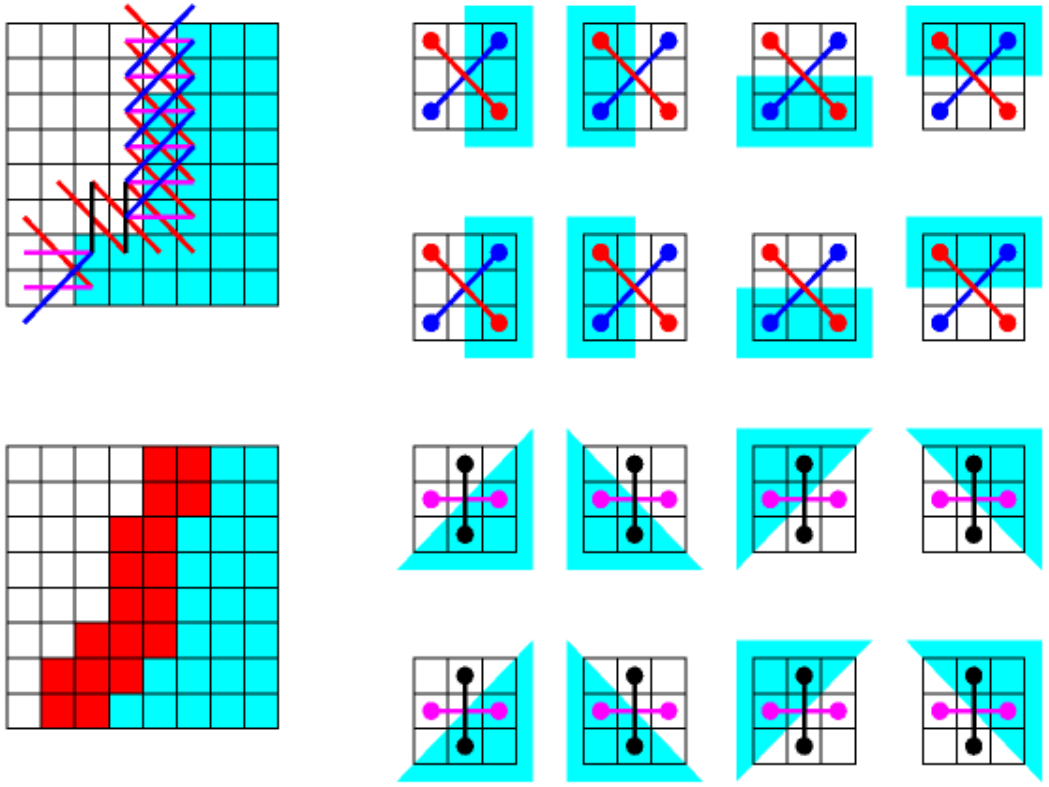




The offset at “two edges and minimum SAD” represents its depth

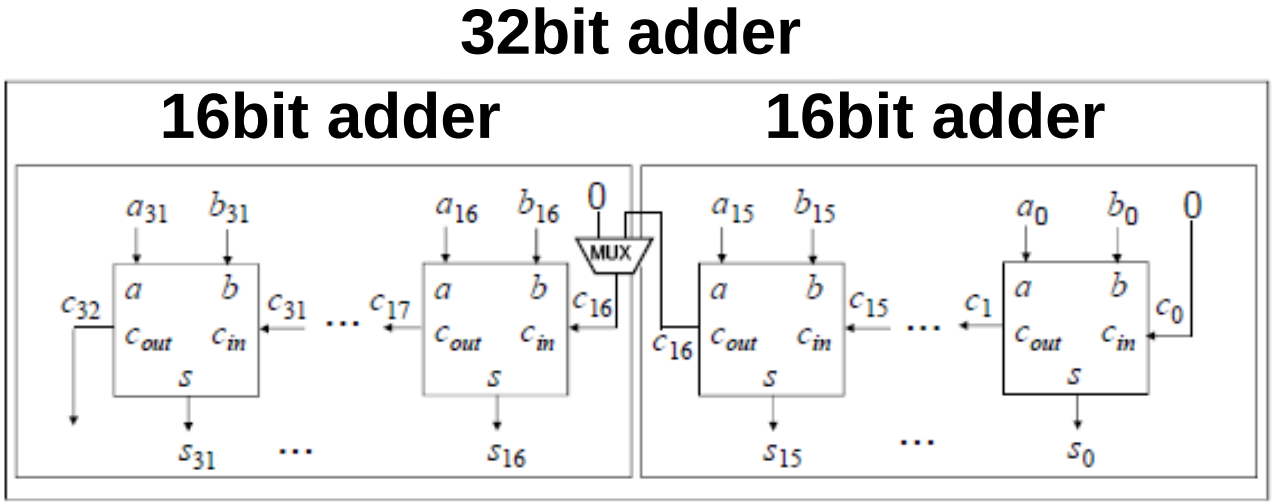
Stereo matching based on edges

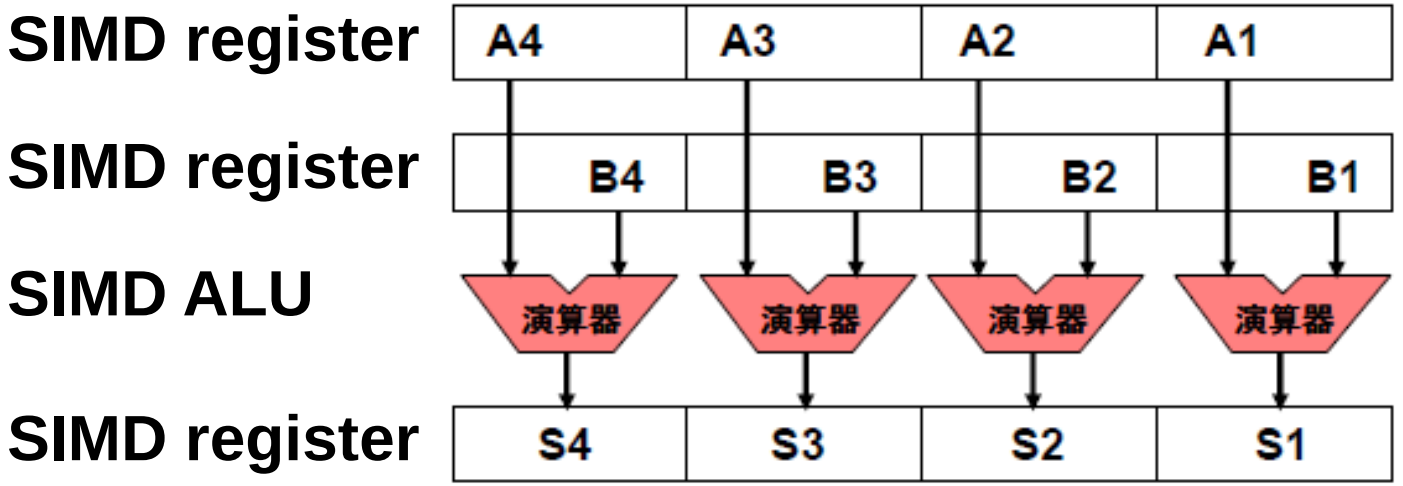
- Edge is detected when SAD with opposite pixels exceeds some threshold.



Speed up with multimedia instructions

Split use of wide registers

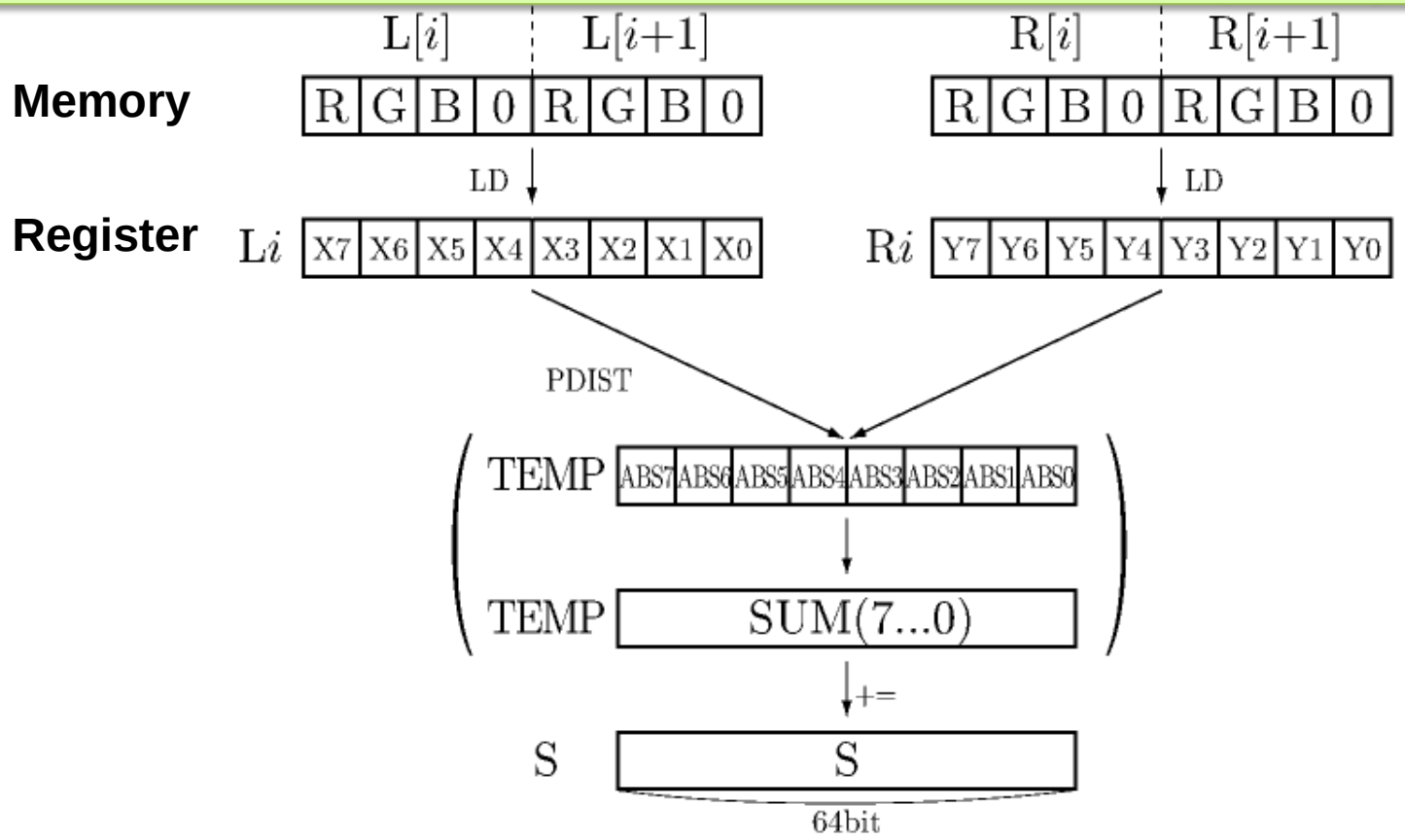




SIMD: Single Instruction and Multiple Data

41

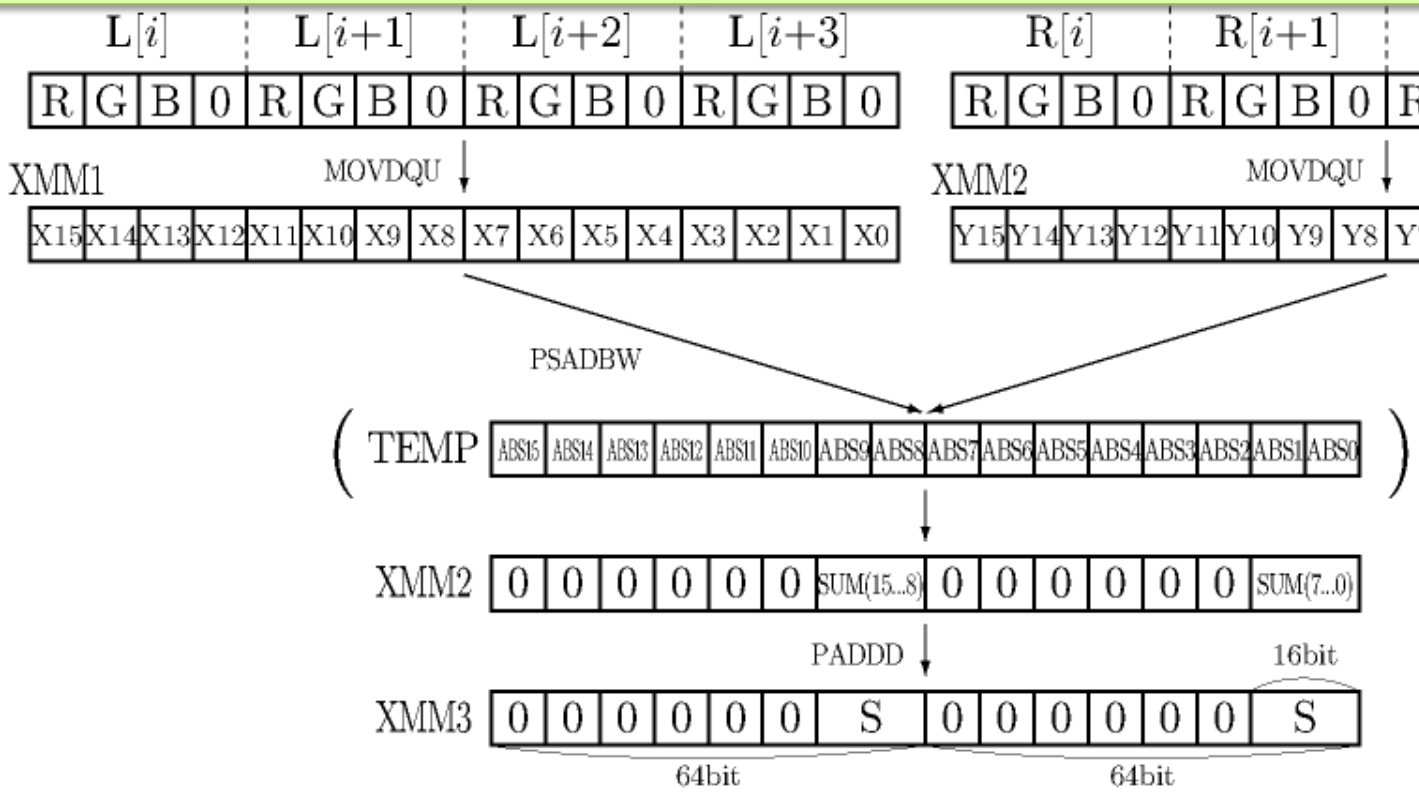
SPARC: Visual Instruction Set (VIS)



- **PDIST: accumulate SAD to 3'rd operand**

```
LD      L[i], L[i+1]  → Li（８バイトロード）
LD      R[i], R[i+1]  → Ri（８バイトロード）
PDIST   Li, Ri, S     → S （８バイトSAD命令）
```

intel: Streaming SIMD Extensions (SSE2)



- **PSADBW: add bytewise SAD (with no accumulation)**

```
MOVDQU L[i],L[i+1],L[i+2],L[i+3] → xmm1 (16B)
MOVDQU R[i],R[i+1],R[i+2],R[i+3] → xmm2 (16B)
PSADBW xmm1, xmm2                → xmm2
PADDD  xmm2, xmm3                → xmm3 (16B)
```

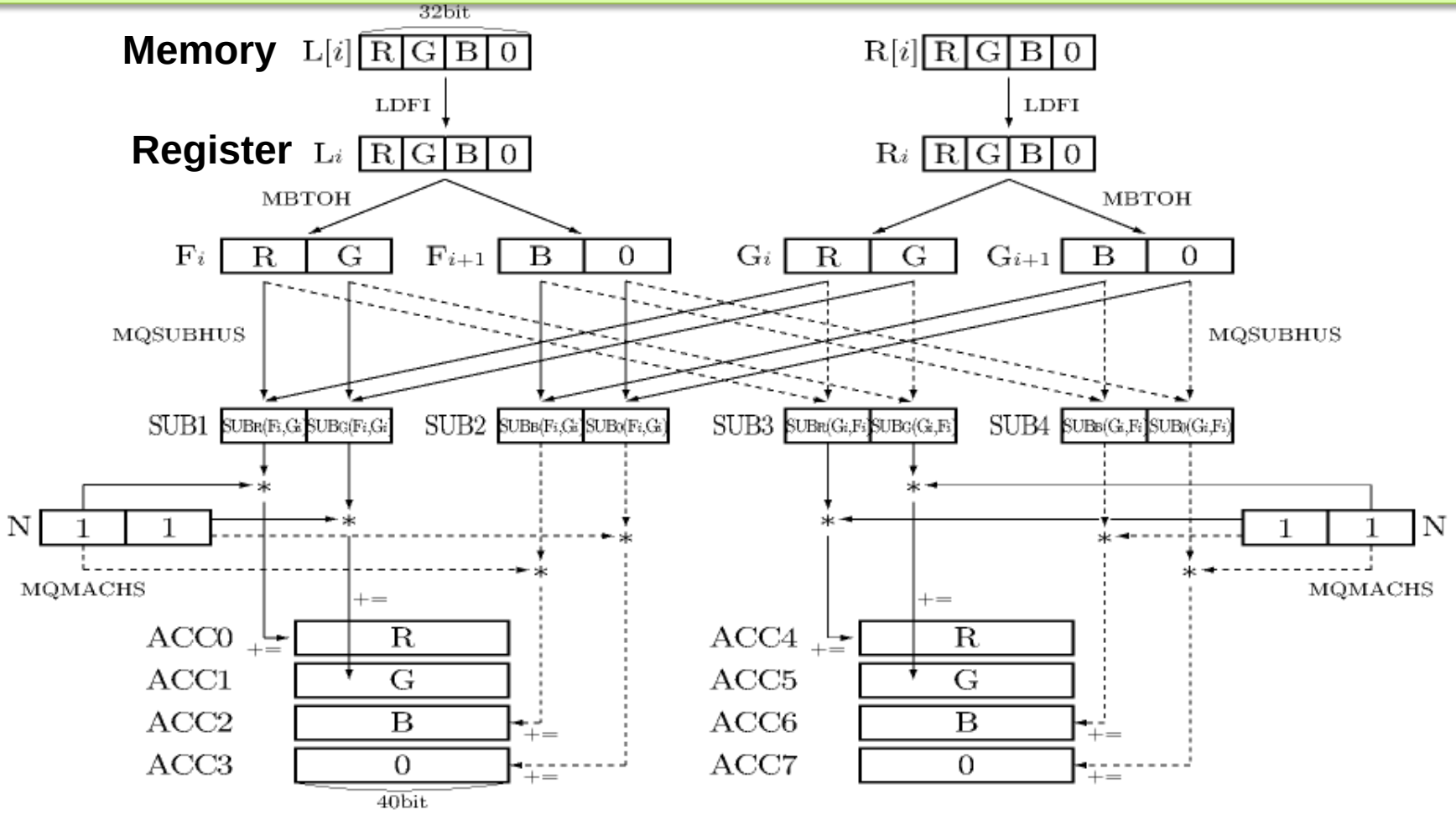
intel: Streaming SIMD Extensions (SSE2)

In case of window: 20x20 pixels

```
wdif:
        pushl     %ebp
        movl      %esp, %ebp
        movl   12(%ebp),%ecx   /* *lp */
        movl   16(%ebp),%edx   /* *rp */
        pxor      %xmm3,%xmm3
        movdqu  0(%ecx),%xmm1  /* line#1 */
        movdqu  0(%edx),%xmm2
        psadbw    %xmm1,%xmm2
        paddd     %xmm2,%xmm3
        movdqu 16(%ecx),%xmm1
        movdqu 16(%edx),%xmm2
        psadbw    %xmm1,%xmm2
        paddd     %xmm2,%xmm3
        movdqu 32(%ecx),%xmm1
        movdqu 32(%edx),%xmm2
        psadbw    %xmm1,%xmm2
        paddd     %xmm2,%xmm3
        movdqu 48(%ecx),%xmm1
        movdqu 48(%edx),%xmm2
        psadbw    %xmm1,%xmm2
        paddd     %xmm2,%xmm3
        movdqu 64(%ecx),%xmm1
        movdqu 64(%edx),%xmm2
        psadbw    %xmm1,%xmm2
        paddd     %xmm2,%xmm3
        addl      $1280,%ecx
        addl      $1280,%edx
        ⋮
        movdqu  0(%ecx),%xmm1  /* line#20 */
        movdqu  0(%edx),%xmm2
        psadbw    %xmm1,%xmm2
        paddd     %xmm2,%xmm3
        movdqu 16(%ecx),%xmm1
        movdqu 16(%edx),%xmm2
        psadbw    %xmm1,%xmm2
        paddd     %xmm2,%xmm3
        movdqu 32(%ecx),%xmm1
        movdqu 32(%edx),%xmm2
        psadbw    %xmm1,%xmm2
        paddd     %xmm2,%xmm3
        movdqu 48(%ecx),%xmm1
        movdqu 48(%edx),%xmm2
        psadbw    %xmm1,%xmm2
        paddd     %xmm2,%xmm3
        movdqu 64(%ecx),%xmm1
        movdqu 64(%edx),%xmm2
        psadbw    %xmm1,%xmm2
        paddd     %xmm2,%xmm3
        movd      %xmm3,%ecx    Get lower 64bits
        psrldq    $8,   %xmm3   Shift right by 8B
        movd      %xmm3,%eax    Get upper 64bits
        addl      %ecx, %eax    Add
        leave
        ret
```

NAIST

FR550: 8way Very long instruction word (VLIW)



Accumulate register

NAIST

FR550: 8way Very long instruction word (VLIW)

ADDL	ADDR			MQSUBHUSSi+8	MQSUBHUSTi+8	MQMACHSSi+1	MQMACHSTi+1
LDFILi	LDFIRi			MQADDHUSX	MQADDHUSZ	MQMACHSSi+2	MQMACHSTi+2
LDFILi+1	LDFIRi+1			MQADDHUSY	MQADDHUSU	MQMACHSSi+3	MQMACHSTi+3
LDFILi+2	LDFIRi+2					MQMACHSSi+4	MQMACHSTi+4
LDFILi+3	LDFIRi+3	MBTOHLi	MBTOHRi			MQMACHSX	MQMACHSZ
LDFILi+4	LDFIRi+4	MBTOHLi+1	MBTOHRi+1			MQMACHSY	MQMACHSU
LDFILi+5	LDFIRi+5	MBTOHLi+2	MBTOHRi+2	MQSUBHUSSi	MQSUBHUSTi		
LDFILi+6	LDFIRi+6	MBTOHLi+3	MBTOHRi+3	MQSUBHUSSi+1	MQSUBHUSTi+1		
LDFILi+7	LDFIRi+7	MBTOHLi+4	MBTOHRi+4	MQSUBHUSSi+2	MQSUBHUSTi+2		
LDFILi+8	LDFIRi+8	MBTOHLi+5	MBTOHRi+5	MQSUBHUSSi+3	MQSUBHUSTi+3		
		MBTOHLi+6	MBTOHRi+6	MQSUBHUSSi+4	MQSUBHUSTi+4		
		MBTOHLi+7	MBTOHRi+7	MQSUBHUSSi+5	MQSUBHUSTi+5		
		MBTOHLi+8	MBTOHRi+8	MQSUBHUSSi+6	MQSUBHUSTi+6		
				MQSUBHUSSi+7	MQSUBHUSTi+7	MQMACHSSi	MQMACHSTi
ADDL	ADDR			MQSUBHUSSi+8	MQSUBHUSTi+8	MQMACHSSi+1	MQMACHSTi+1
LDFILi	LDFIRi			MQADDHUSX	MQADDHUSZ	MQMACHSSi+2	MQMACHSTi+2
LDFILi+1	LDFIRi+1			MQADDHUSY	MQADDHUSU	MQMACHSSi+3	MQMACHSTi+3
LDFILi+2	LDFIRi+2					MQMACHSSi+4	MQMACHSTi+4
LDFILi+3	LDFIRi+3	MBTOHLi	MBTOHRi			MQMACHSX	MQMACHSZ
LDFILi+4	LDFIRi+4	MBTOHLi+1	MBTOHRi+1			MQMACHSY	MQMACHSU
LDFILi+5	LDFIRi+5	MBTOHLi+2	MBTOHRi+2	MQSUBHUSSi	MQSUBHUSTi		
LDFILi+6	LDFIRi+6	MBTOHLi+3	MBTOHRi+3	MQSUBHUSSi+1	MQSUBHUSTi+1		
LDFILi+7	LDFIRi+7	MBTOHLi+4	MBTOHRi+4	MQSUBHUSSi+2	MQSUBHUSTi+2		
LDFILi+8	LDFIRi+8	MBTOHLi+5	MBTOHRi+5	MQSUBHUSSi+3	MQSUBHUSTi+3		

MQMACHS: quad 16bit-multiply-and-add
Moreover, 8 instructions can be executed simultaneously.

NAIST