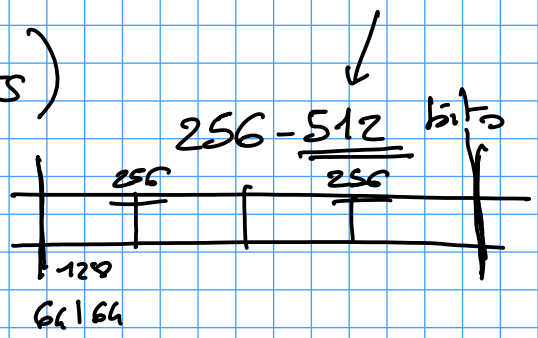
VECTORIZATION

Vector units (few units)

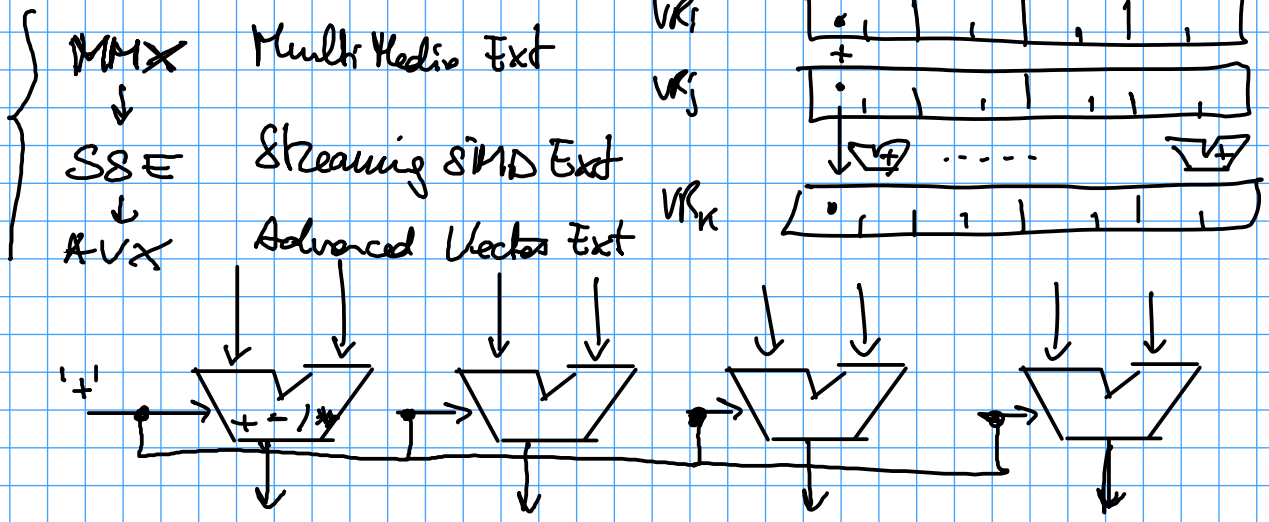
Vector register file

Vector ALUs

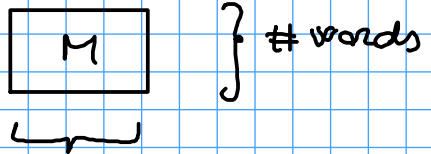


8 × 64

ADD^VL VRi, VRj, VRk

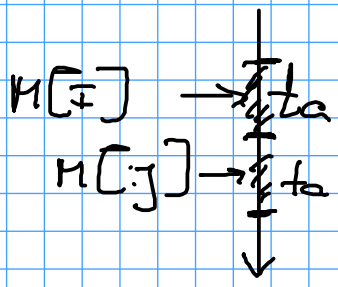


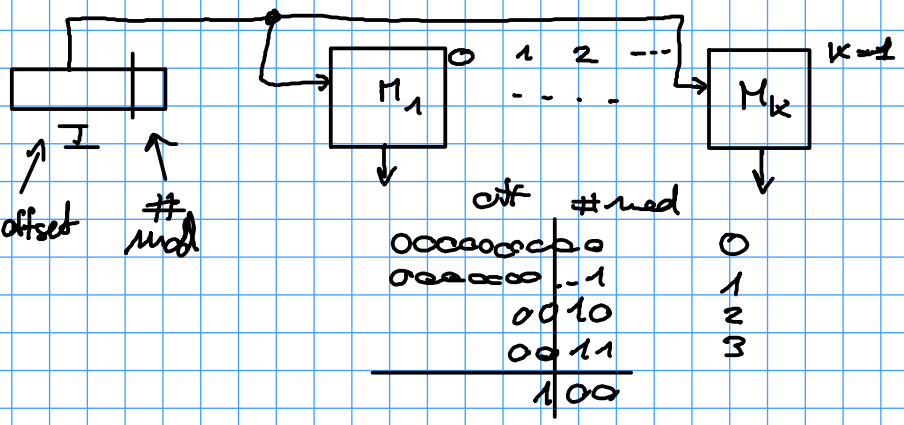
Vector LOAD / STORE



64

32bit
64 bits





∀Mi 1M words of 64b

Load / store e vector registers in $\frac{\#\text{items} \times VR}{K}$

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

int64

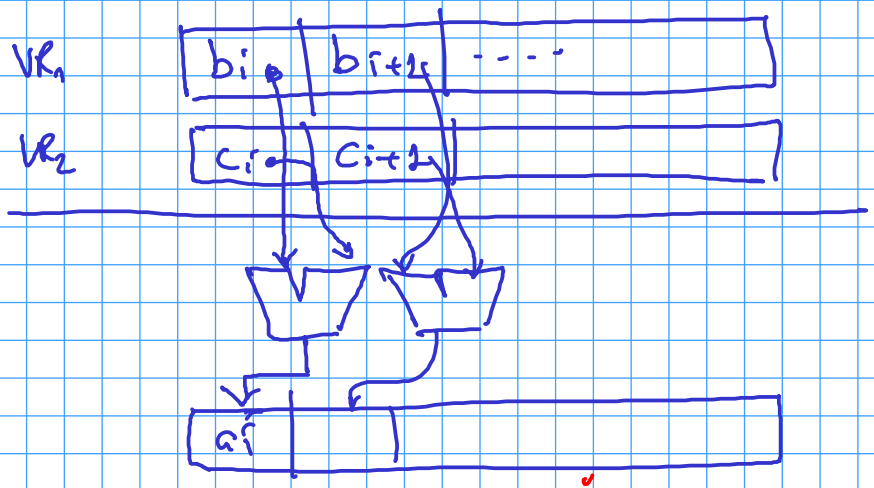
```
ta   LOAD  ...  b[i]   ta
ta   LOAD  ...  c[i]   ta
ta   ADD   ...  a[i]
ta   STORE ...  a[i]   ta
ta   INC   Ri
ta   test and loop condition
     fetch
```

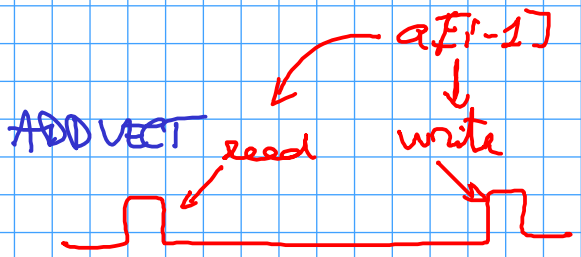
NON VECTOR CODE

VECTOR CODE

```
ta   LOADVEC   VR1 ← b[i], b[i+1] ... b[i+7]   ta
ta   LOADVEC   VR2                              ta
ta   ADDVEC                                     ta
ta   STOREVEC
ta   INC  +8
ta   test
```

4 iterations of the loop






```
for(i=1; i<N; i++)
    a[i] = a[i-1] + b[i];
```

Loop carried dependencies

dependencies { RAW (read after write) I_i ; I_{i+1} $\quad R(I_{i+2}) \cap W(I_i) \neq \emptyset$

& places where I read and write x in one different

WAR (this may be ignored in case of synchronous operations (same x))

RAR

WAW

RAR ⇒ Multiple reads of the same value

WAW ⇒ x = 1; x = 2

both may be somehow ignored

Loop carried dependency

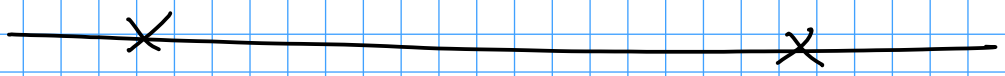
↓

a dependency

$I_{i,i} \rightarrow I_{i+1,k}$

I_i from iteration x

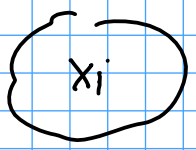
$I_{i+1,k}$ " " x+1



Alignment of data accesses

auto a = new float[n][n];

__declspec(align(16,0)) float a[128];



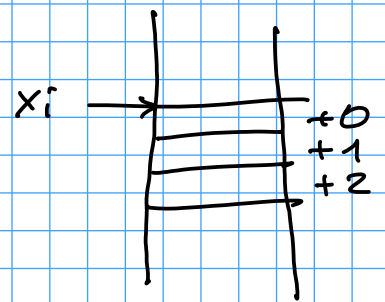
R
G
B
} color

```
struct { int R;
         int G;
         int B; } [N]
```

for(...){

x[i].R = ...

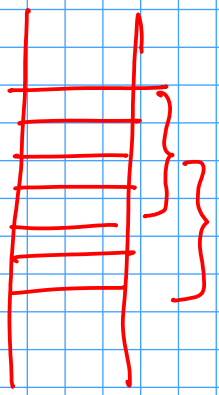
x[i].B = ...



```
struct { int R[N];
         INT A[N];
         int B[N]; }
```

X[i].G

```
... foo( float * a, float *b, int n ) {
       for( .... )
         a[i] = b[i]+1;
     }
```



=restricted float *a;

usage of functin calls

```
for(i=0; i<N; i++)
  x[i] = fun(y[i]);
```

default loops with fun calls are not vectorized

makes { fun is inlined
fun is one of the intrinsics

#of iterations

```
for( --- )
while( ) {}
```

✓ not ok

- no "break" exits the loop
- should not use non constant loop its boundaries

#pragma simd

#pragma ivdep