CS 451 / 551 / ECE 541

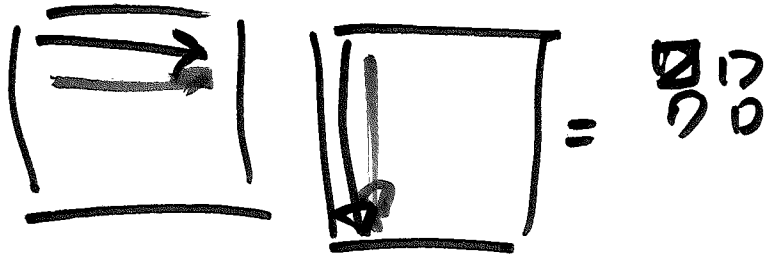
ADVANCED
COMPUTER ARCHITECTURE

SESSION no. 20

University of Idaho

VECTOR PROCESSORS

- SIMD ← "STREAM", MATRIX - VECTOR OPS
  PROBLEM - EXPRESS IN TERMS OF VECTOR OPERATIONS - "VECTORIZE"
  - N-DIM MATRIX OPS - DECOMPOSE INTO SERIES OF 1-D OPS.



VECTOR REGISTER: HOLD VECTORS

VMIPS - 64 ELEMENTS x 64 BITS

- DOUBLE PRECISION F.P.
- LONG INT
- SMALLER FIELDS
  - 2 x 32 BIT
  - 4 x 16 BIT SHORT
  - 8 x 8 BIT BYTES

VECTOR OPS

ONE INSTR ⇒ 64 COPIES OF THE OP

ADDVV
|
VECTOR-TO-VECTOR

ADDVS
|
VECTOR-TO-SCALAR

ADVANTAGES OF VECTOR OVER SCALAR?

1. FASTER
   - MINIMIZE FETCH & DECODE
   - OPTIMIZE HARDWARE FOR SPEED (RESTRICTED INSTRUCTION SET)

2. POWER SAVINGS

CONVOY - STACK INSTRUCTIONS, INPUT TO PIPELINE.
- INDEPENDENT (NO DEPENDENCIES)

University of Idaho

ENHANCEMENTS - PERFORMANCE & FLEXIBILITY

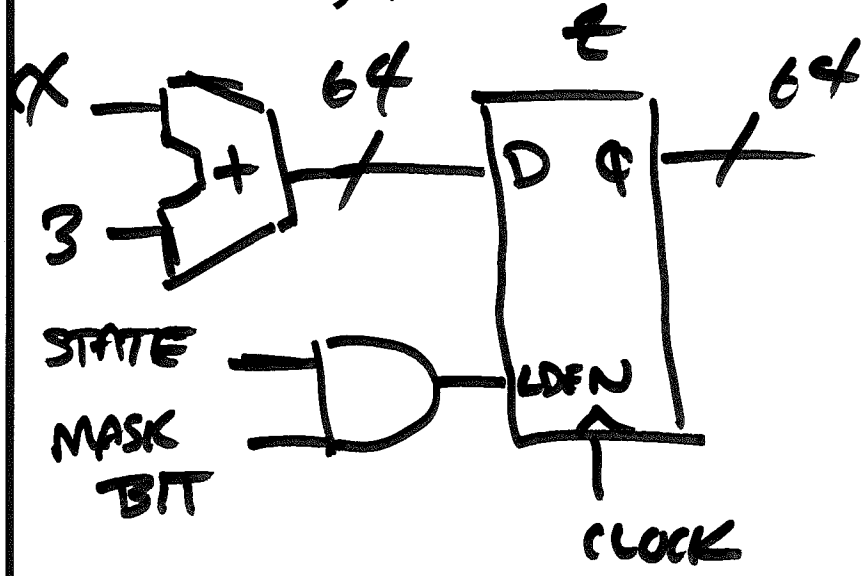
1. MULTIPLE ELEMENTS / CLOCK CYCLE
   - MULTIPLE EXECUTION UNITS

*2. MULTIPLE VECTOR LENGTHS
   ADD LENGTH REGISTER

3. IF STATEMENTS

```
for (_____)   <- vector x 64
  if (x=y)    x, y vectors
    z = x + 3;
```

x	2	3	[3]	9	[7]	-2	
y	1	4	[3]	6	[7]	-3	
Mask	0	0	1	0	1	0	0 0



4. MEMORY BW
5. MULTIPLE DIMENSIONS (MATRICES)
6. SPARSE MATRICES — (STRIDE)
   INDEX VECTOR
7. PROGRAMMING

University of Idaho

MULTIPLE LENGTH VECTORS

DATA VECTOR LENGTH ≠ VECTOR REGISTER LENGTH

MVL = Max Vector length

VL = vector length (odd size piece)

n = size of input data

ex MVL = 16

n = 87

VL = 87 % 16 = 7

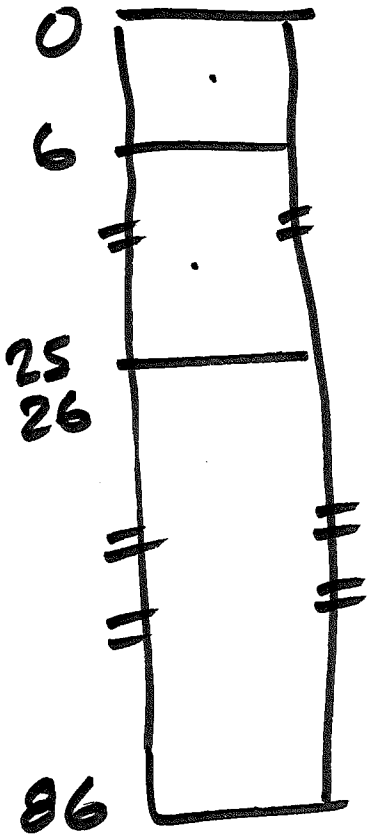
$16 \overline{)87}$ R 7

TO REPRESENT 87 WORDS

5 16-BIT VECTORS

1 7-BIT VECTOR

```
low = 0
VL = n % MVL -  START WITH ODD (SHORT) SEGMENT = 7
for (j=0; j<=(n/MVL); j++)
    for (i=low; i < low + VL ; i++){
        Y[i] = a * X[i] + Y[i]
    low = low + VL;
    VL = MVL   - USE LONGER SEGMENT =16 FOR REMAINDER (5 TIMES)
}
```



IN HARDWARE

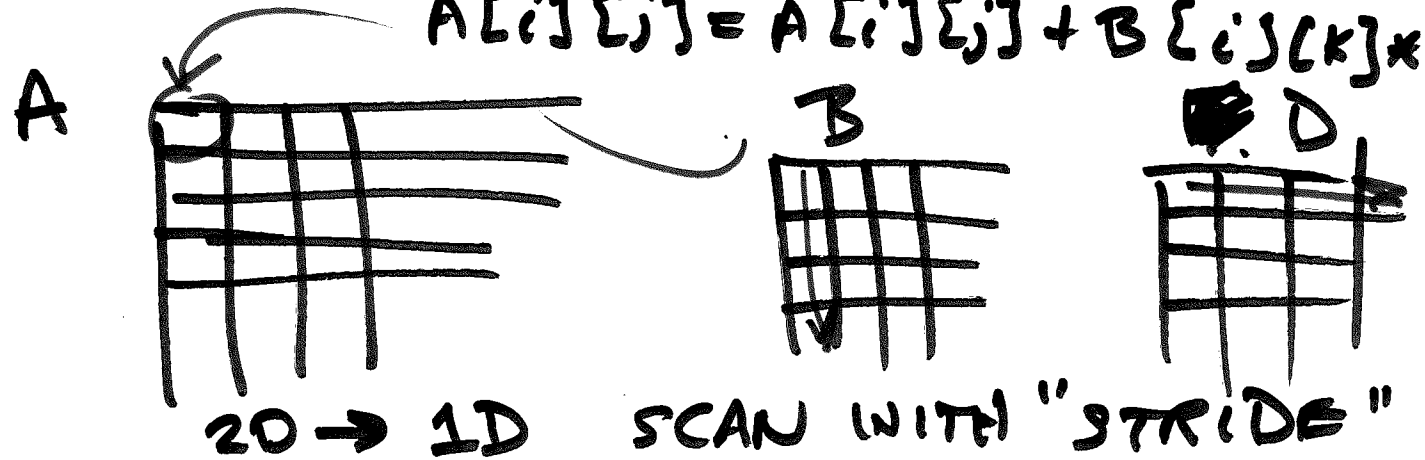
- VECTOR LENGTH REGISTER
  - INITIALLY - SET TO 7
    ONLY DO 7 OPS.
  - THEN SET TO 16

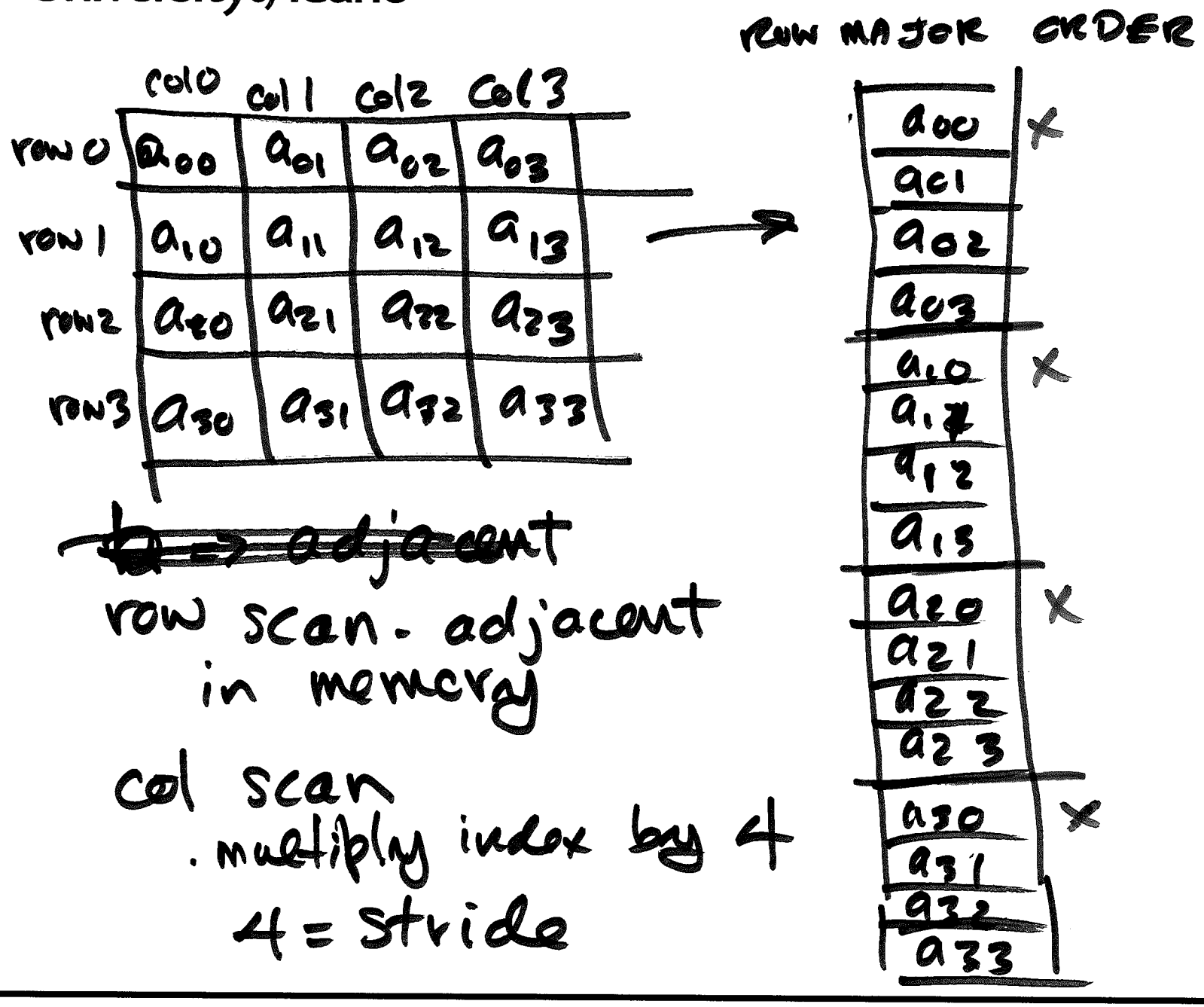
~~5. MULTIDEI MULTIDE~~

5. MULTIDIMENSIONAL ARRAYS

e.g. MATRIX MULTIPLY

```
for(i=0; i<100; i++)  // ∀ row
  for(j=0; j<100; j++)  //∀ column
  A[i][j] = 0.0
  for(k=0; k<100; k++)
    A[i][j] = A[i][j] + B[i][k] * D[k][j]
```





~~a => adjacent~~

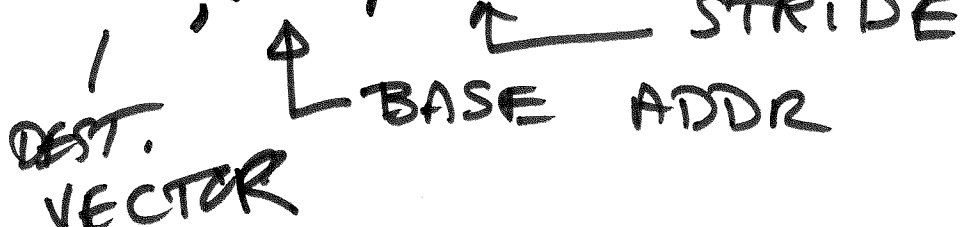
row scan. adjacent in memory

col scan
. multiply index by 4
4 = stride

NEED ABILITY TO LOAD VECTORS

- ONE WITH ADJACENT MEMORY LOCATIONS (ROW SCAN)
- ONE WITH "STRIDE" (COL SCAN)

LVWS V1, (R1, R2)



V1 = MEM[R1], MEM(R1+R2],
MEM[R1 + 2·R2]

GATHER NON CONTIGUOUS MEMORY INTO CONTIGUOUS VECTOR

PUSH VECTORS THRU PIPELINE

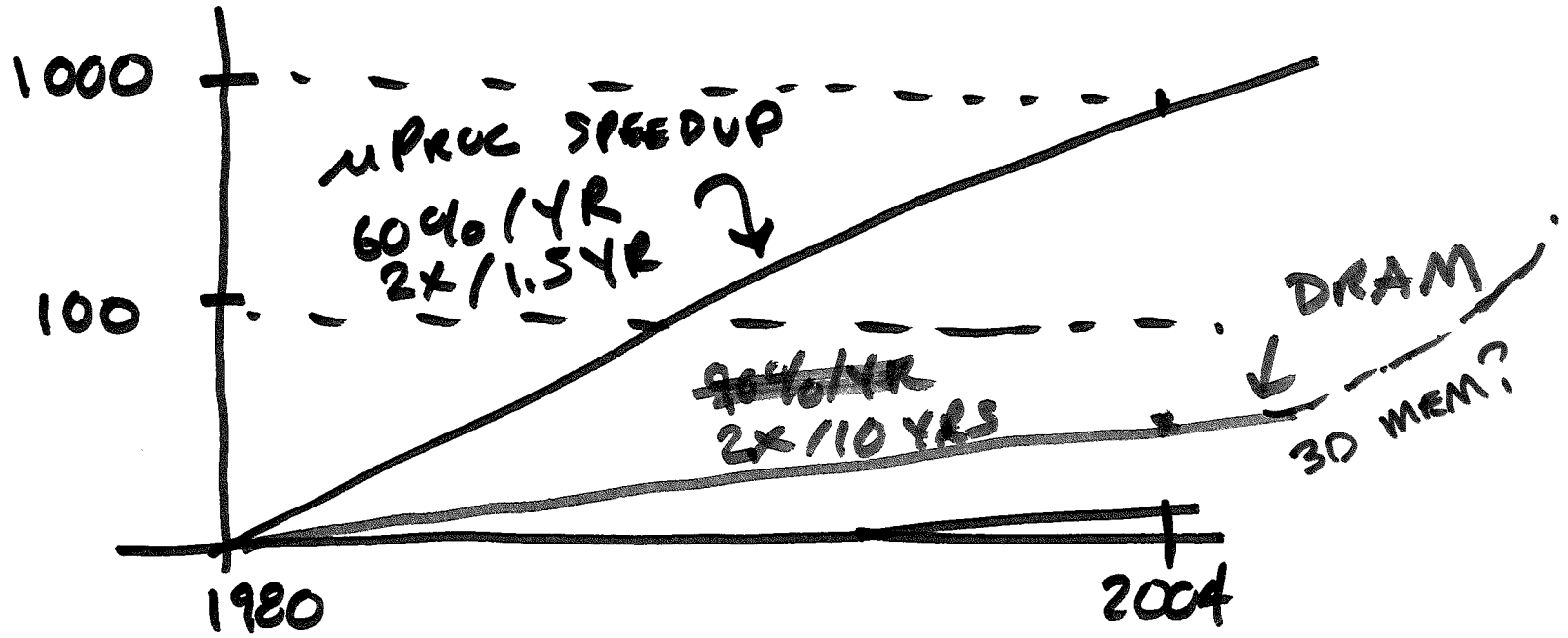
University of Idaho

SVWS - INVERSE

"STORE VECTOR WITH STRIDE"

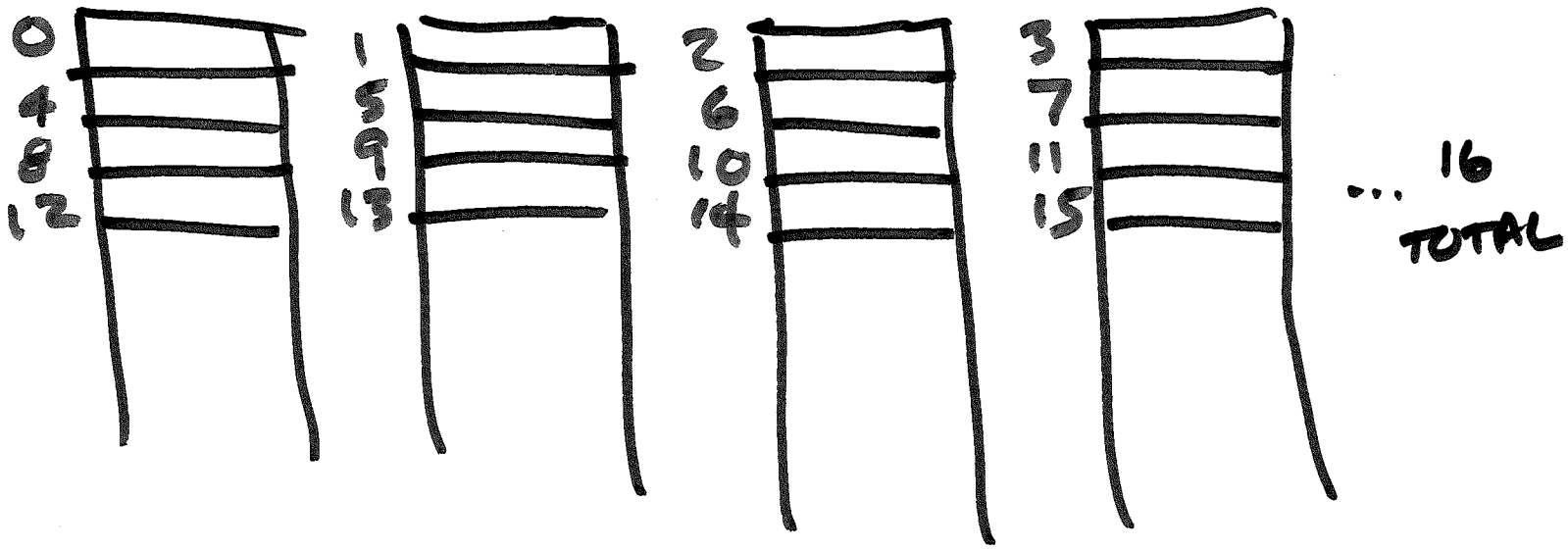
---

4. MAKE MEMORY BW MATCH PROCESSOR SPEED



MEMORY BANKS

- CONTIGUOUS ADDRESS SPACE
- SEPARATE PHYSICAL MEMORIES



- READ OR WRITE VECTORS – EACH VECTOR ELEMENT FROM A DIFFERENT "MEMORY BANK." IN PARALLEL.
- AMORTIZE ACCESS LATENCY

University of Idaho

EXAMPLE P.277

Q: HOW MANY "BANKS" DO I NEED TO KEEP PROCESSOR BUSY?

CRAY T932

32 PROCESSORS

4 LOADS, 2 STORES / CLOCK CYCLE

2.167 ns / CLOCK CYCLE

15 ns MEM CYCLE TIME BETWEEN MEMORY ACCESSES

$32 \text{ PROC.} \times 6 \frac{\text{ACCESSES}}{\text{PROC.}} = 192 \frac{\text{ACCESSES}}{\text{CYCLE}}$

$\frac{15}{2.167} = 7 \text{ CYCLES/ACCESS}$

$192 \times 7 = 1344$ MEMORY BANKS!

HAS 1024 MEMORY BANKS