Convergence

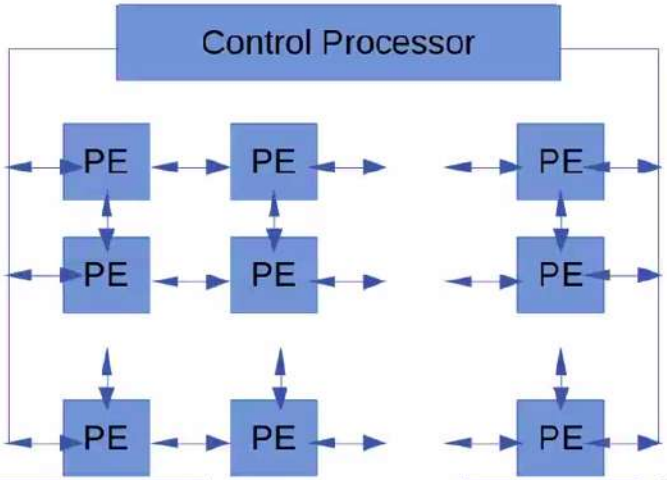
- Various ways to support shared memory on MPI machines and MPI on shared memory machines
- Evolution and role of software has blurred the boundary
  - send/recv supported on shared address space machines using buffers
  - Global address space available
- Now the underlying machine structure has converged towards a common organisation + communication assist / infrastructure
- Hardware organisation also converging
  - Tigher integration of network interface with the cores gives low latency and high bandwidth
- Even clusters of workstations/SMPs are parallel systems
  - These depend on fast system area network

Other types of Parallel Architectures

Data Parallel Processing

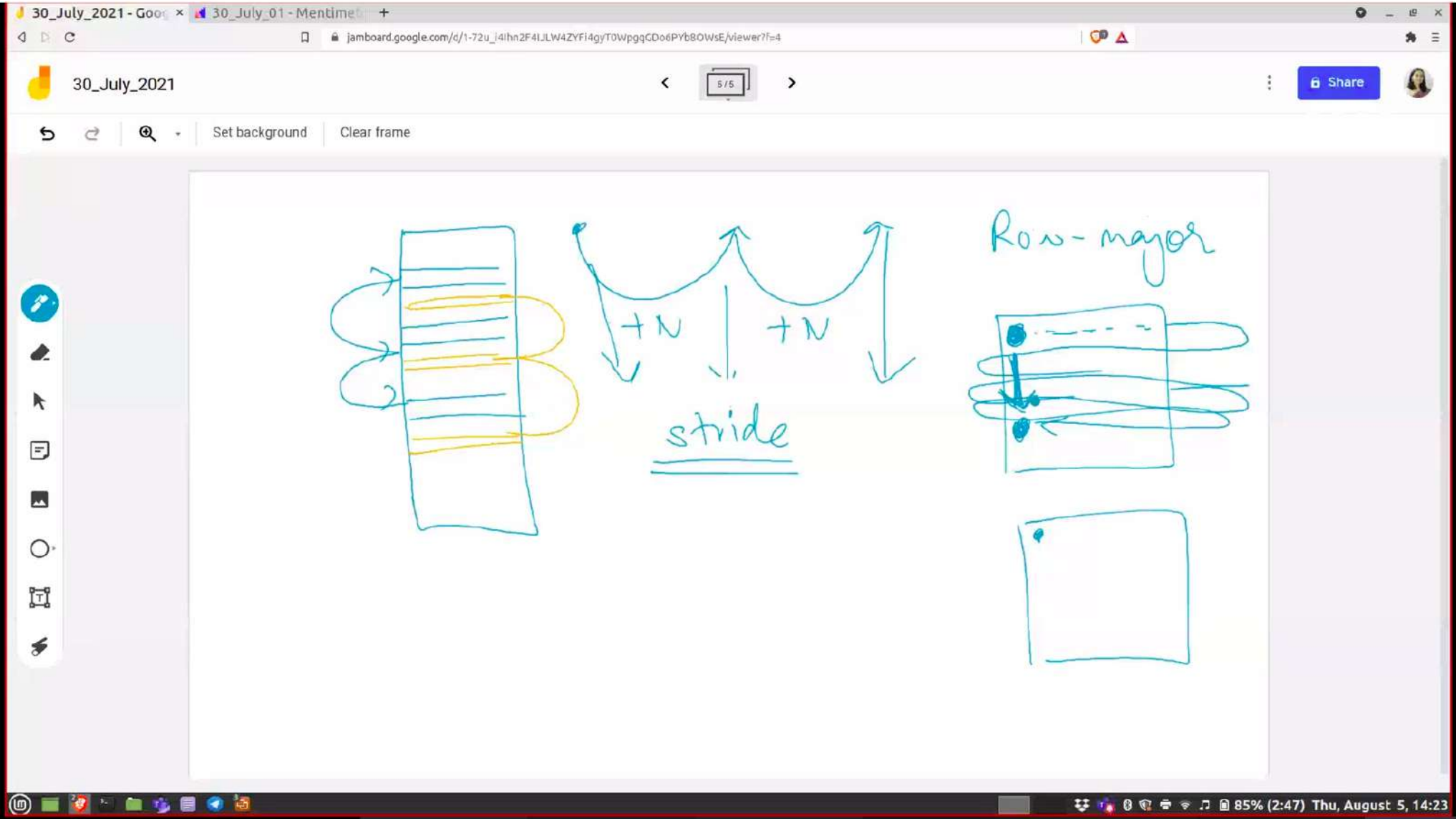
- Here, operations can be performed in parallel on each element of a large regular data structure such as an array or matrix
- Gave rise to Flynn's Taxonomy that categories designs in terms of number of distinct instructions issued at a time and the number of data elements they operate on
- SISD: old style processor
- SIMD: like GPUs
- MIMD: shared memory,MPI, multi-programmed etc.
- MISD: no meaning

SIMD = data parallelism
Same instruction on different data

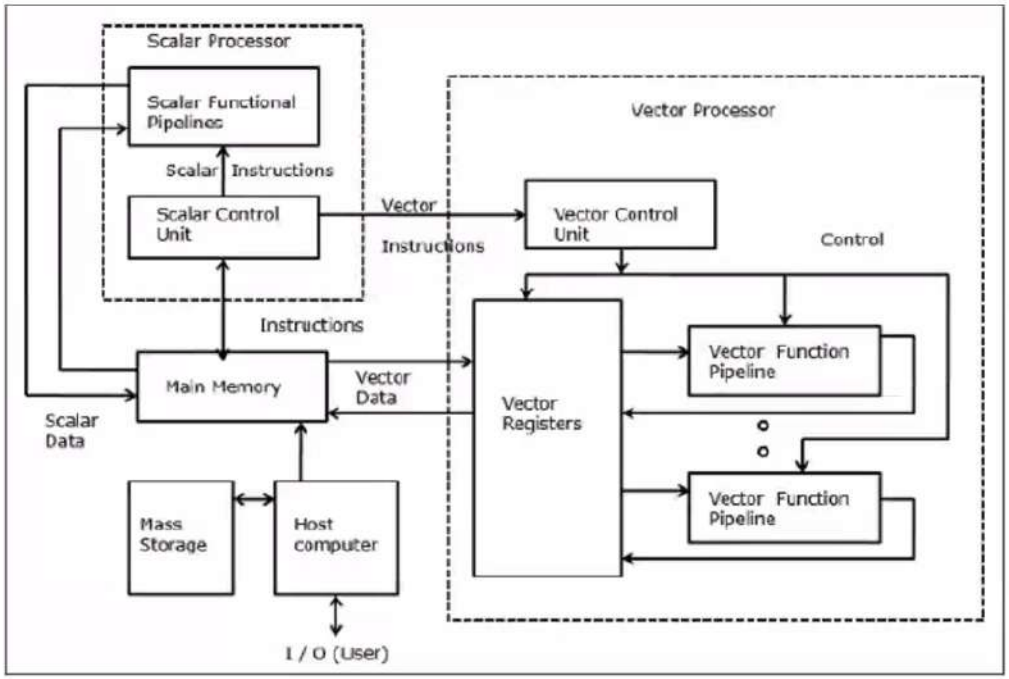


Vector Processors

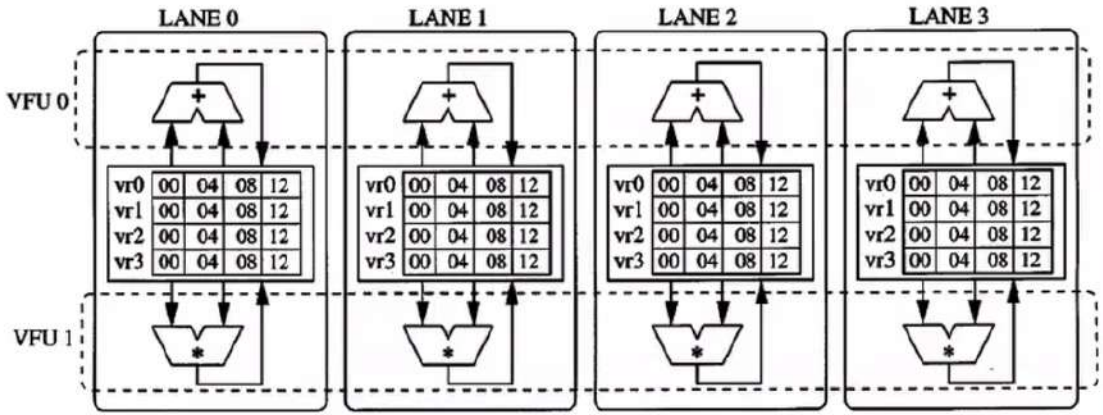
- These operate on arrays or vectors of data
- CPUs operate on individual data elements or scalars
- Recent systems have
  - Vector registers: length of reg=4 to 128, 64-bit elements
  - Vectorised and pipelined functional units
  - Vector instruction
  - Interleaved memory
  - Strided memory access and hardware scatter/gather
    - Access 1st, 5th, 9th element
    - (Scatter = write, Gather = read) at regular intervals (1st, 2nd, 4th, 8th element)


30_July_2021 - Goog
30_July_01 - Mentimet
jamboard.google.com/d/1-72u_i4Ihn2F4IJLW4ZYFi4gyT0WpgqCDo6PYb8OWsE/viewer?f=4
30_July_2021
5/5
Share
Set background
Clear frame
+N
+N
stride
Row-major
85% (2:47) Thu, August 5, 14:23
+33
AG
DS
SUBRATA ROY
NISHANK SIDDHARTH
NS
NISHTHA SHARMA
DG
DEVANSHI GUPTA
VA
VARHADE AMEY ANANT
A
AMIT
Hemangee Kalpesh Kapoor

Vector Processors



Vector Processors




```
# C code
for (i=0; i<64; i++)
  C[i] = A[i] + B[i];
```

```
# Scalar Code
  LI R4, 64
loop:
  L.D F0, 0(R1)
  L.D F2, 0(R2)
  ADD.D F4, F2, F0
  S.D F4, 0(R3)
  DADDIU R1, 8
  DADDIU R2, 8
  DADDIU R3, 8
  DSUBIU R4, 1
  BNEZ R4, loop
```

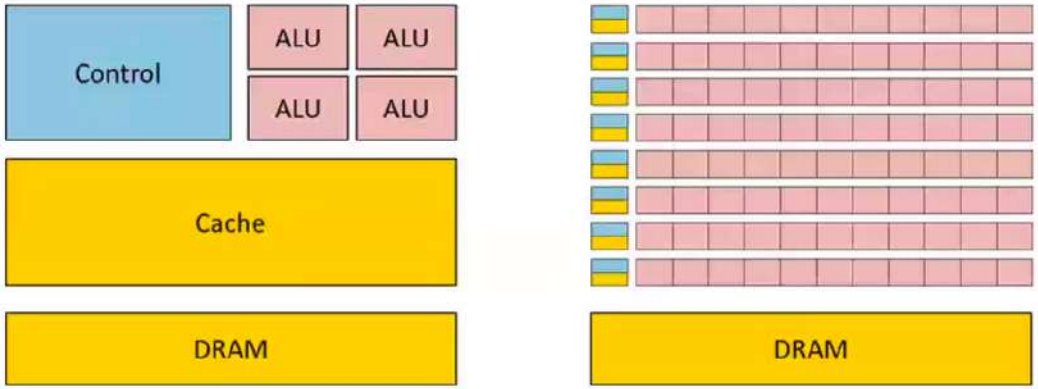
```
# Vector Code
  LI VLR, 64
  LV V1, R1
  LV V2, R2
  ADDV.D V3, V1, V2
  SV V3, R3
```

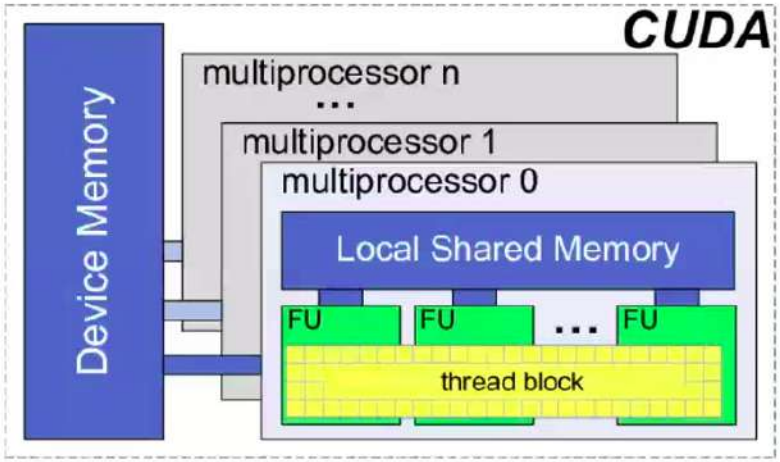
Vector Processors

- For matrix multiplication: A(i,j) = A(i,j) + B(i,k) * C(k,j)
  - B or C accesses will not be to adjacent or consecutive locations (800 bytes apart)
  - Stride: distance separating the elements that are to be merged into a single vector
  - Caches do unit stride
  - LVWS (load vector with stride) instruction
  - SVWS (store vector with stride)
  - Strides can cause bank conflicts and a stall may occur
- Vector systems have special hardware to accelerate strided access
- Fast and easy to use
- Have high memory bandwidth
- BUT cannot handle irregular data structures and have finite limit on scalability. To handle ever larger problems => larger vectors !

Graphics Processing Units (GPUs)

- Real-time graphics APIs use points, lines and triangles to internally represent the surface of an object
- They use graphics processing pipelines to convert the internal representation into an array of pixels to be sent to the screen
- The working of the graphics pipeline stages is specified by shader functions
- Shader functions are very short (few lines) and several of them can be executed in parallel
- GPUs have large number of ALUs on each GPU core
- Processing single image requires 100s of MB of data

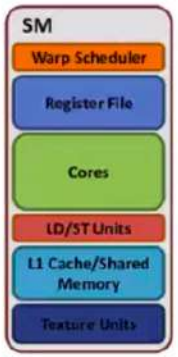




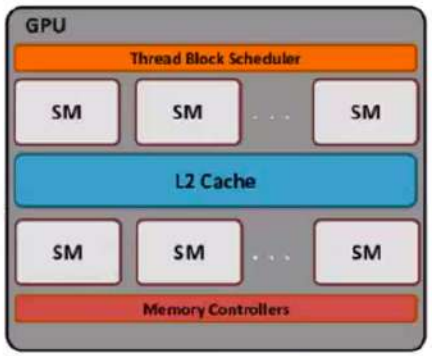
GPUs

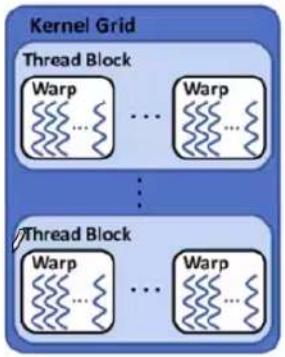
Other archictectures
Are
Dataflow
And
systolic

- GPUs have
  - High data rates
  - Hardware multi-threading
  - Store state of hundred suspended threads
- We need lots of work to keep the GPUs busy !
- SIMD but some GPUs do multi-program



Streaming multiprocessor





Shared Memory Multiprocessors

Shared Memory

Prevalent Parallel Architecture

- The most prevalent parallel architecture is multiprocessor of small to moderate scale that provides
  - a global physical address space + symmetric access to all of the main memory from any procesor (SMP = Symmetric Multi Processor)
- Each processor has its own cache
- All processors are connected over a common interconnect
- These also form building blocks for larger scale system
- Efficient sharing of resources like CPU + Main Memory enable their use as
  - Throughput Engines
  - Attractive for Parallel Programming
  - Useful for Operating Systems

Prevalent Parallel Architecture

- The most prevalent parallel architecture is multiprocessor of small to moderate scale that provides
  - a global physical address space + symmetric access to all of the main memory from any procesor (SMP = Symmetric Multi Processor)
- Each processor has its own cache
- All processors are connected over a common interconnect
- These also form building blocks for larger scale system
- Efficient sharing of resources like CPU + Main Memory enable their use as
  - Throughput Engines
  - Attractive for Parallel Programming
  - Useful for Operating Systems

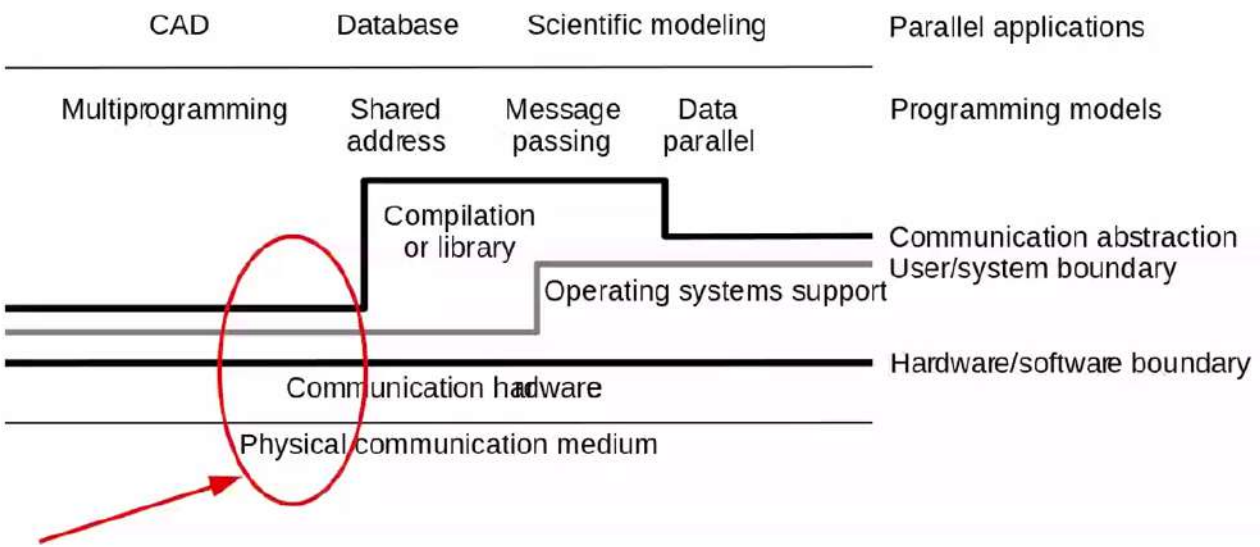
Why is shared memory useful?

- Throughput engine
  - For multiple sequential jobs with varying memory and CPU requirements
- Useful for parallel programming
  - As they provide ability to access any memory location from any processor using ordinary LOAD/STORE
  - Automatic movement and replication of shared data in the local caches
- Useful for OS
  - As the different processes of OS shared data structures and easily run on different processors

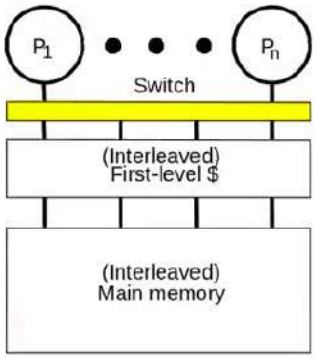
Why is shared memory useful?

- User processes read/write shared virtual address
- Realised as LOAD/STORE on shared physical address
- On the contrary, Message passing needs support from intermediate software layer (i.e. run-time library)
- Message passing buffers treated as shared memory
- Communicate + compute generate memory accesses in shared address space

Layer Perspective

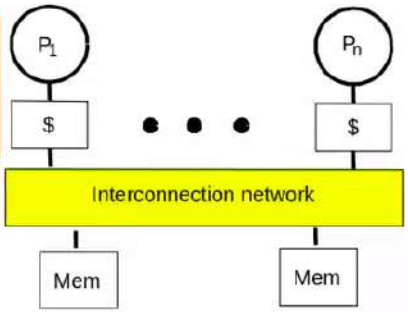


Natural extensions of memory system

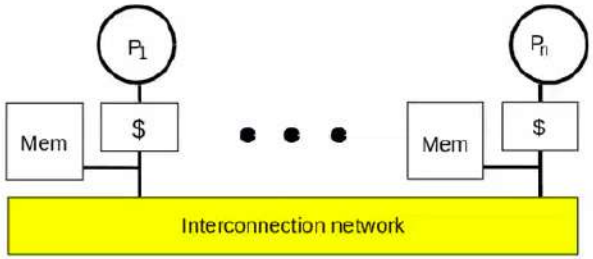


Shared Cache

Bus Based designs



Centralized Memory
Dance Hall, UMA



Distributed Memory (NUMA)