Lecture 28-29: October 28, 2020

Computer Architecture and Organization-II

Biplab K Sikdar

Parallel Computers -III

There are many other interconnection network proposed for SIMD machines. Selection of network depends on application for which SIMD machine is used.

Example 0.3 *Consider addition of n numbers.*

Best solution can be - execution in a tree connected SIMD m/c.

Summation can be resulted in O($\log n$) time.

Follow the addition of 8 numbers 1, 2, 2, 3, 4, 3, 5 and 6 in Figure 25.

Processor placed at root node position stores final sum.

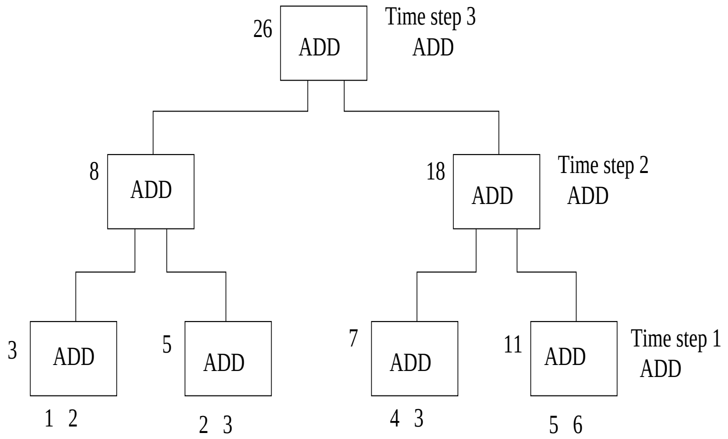


Figure 25: Addition of 8 numbers in tree connected SIMD macine

Matrix transpose

$$A = \begin{vmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{vmatrix} \qquad A^T = \begin{vmatrix} a_{00} & a_{10} & a_{20} & a_{30} \\ a_{01} & a_{11} & a_{21} & a_{31} \\ a_{02} & a_{12} & a_{22} & a_{32} \\ a_{03} & a_{13} & a_{23} & a_{33} \end{vmatrix}$$

Figure 26: Matrix transpose

Solution 1: With mesh of processors

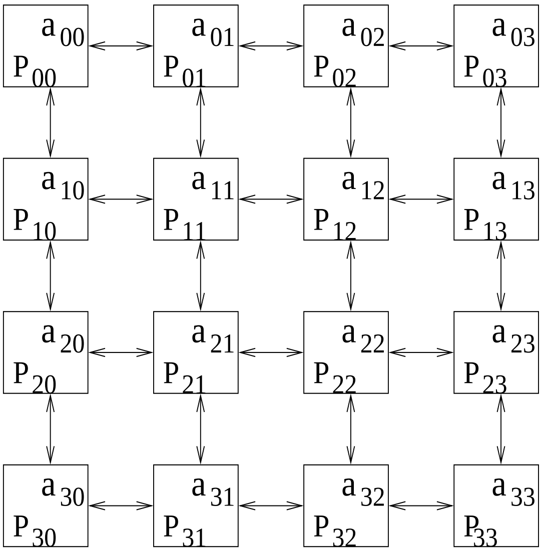


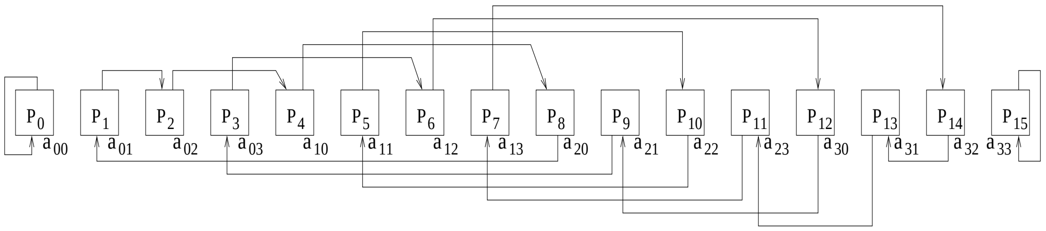
Figure 27: Matrix transpose with mesh network

Here the lower bound of run time is $\omega(n)$.

Because $a_{0,n-1}$ (a_{03}) cannot reach P(n-1,0) (P30) in fewer than $2n$-2 steps.

Matrix transpose

Solution 2: With perfect-shuffle network of processors


P_0
P_1
P_2
P_3
P_4
P_5
P_6
P_7
P_8
P_9
P_{10}
P_{11}
P_{12}
P_{13}
P_{14}
P_{15}
a_{00} a_{01} a_{02} a_{03} a_{10} a_{11} a_{12} a_{13} a_{20} a_{21} a_{22} a_{23} a_{30} a_{31} a_{32} a_{33}

Initial distribution

a_{00} a_{01} a_{02} a_{03} a_{10} a_{11} a_{12} a_{13} a_{20} a_{21} a_{22} a_{23} a_{30} a_{31} a_{32} a_{33}

First shuffle

a_{00} a_{20} a_{01} a_{21} a_{02} a_{22} a_{03} a_{23} a_{10} a_{30} a_{11} a_{31} a_{12} a_{32} a_{13} a_{33}

Second shuffle

a_{00} a_{10} a_{20} a_{30} | a_{01} a_{11} a_{21} a_{31} | a_{02} a_{12} a_{22} a_{32} | a_{03} a_{13} a_{23} a_{33}

Transpose

$$\begin{array}{cccc} a_{00} & a_{10} & a_{20} & a_{30} \\ a_{01} & a_{11} & a_{21} & a_{31} \\ a_{02} & a_{12} & a_{22} & a_{32} \\ a_{03} & a_{13} & a_{23} & a_{33} \end{array}$$

Figure 28: Matrix transpose with perfect shuffle network

For $n \times n$ matrix, after exactly q-shuffle ($n = 2^q$) transpose operation is completed.

Lower bound of run time is $O(logn)$

0.4 MISD computers

MISD machine is designed with N processors each of which is having its own control unit but all processors share a common memory unit.

Processors P_0, P_1, $\cdots$, P_i, $\cdots$, P_{N-1} execute instruction streams (I_{10}, I_{20}, $\cdots$, I_{k0}), (I_{11}, I_{21}, $\cdots$, I_{k1}), $\cdots$ ($\mathrm{I}_{1,N-1}$, $\mathrm{I}_{2,N-1}$, $\cdots$, $\mathrm{I}_{k,N-1}$) respectively.

Shared memory unit is for storing of data accessed by the processors.

During program execution, a set of operands R_{i1}, R_{i2}, $\cdots$, R_{il} is received from memory by P_j to process instruction I_{ij}.

Parallelism is achieved in an MISD computer.

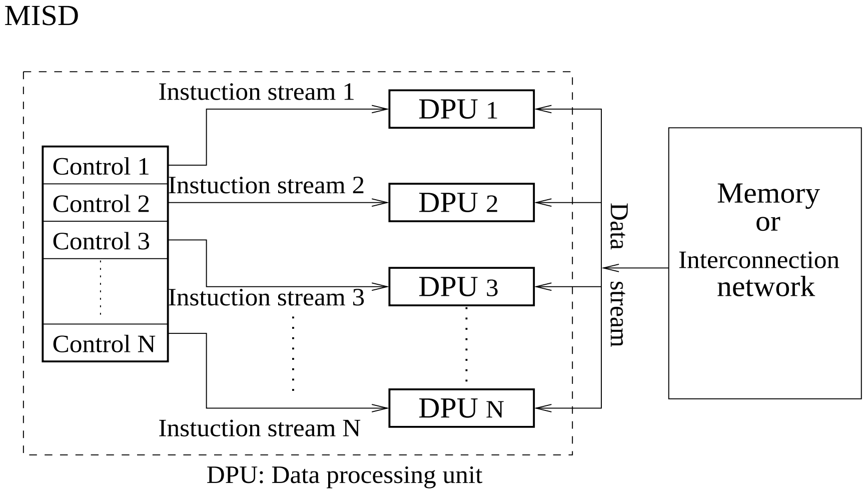


Figure 29: MISD machine

Example 0.4 Consider brute force solution to check whether a positive number M is prime. Therefore, the task is -

M should be divided by 2, 3, 5, 7, $\cdots \lfloor\sqrt{N}\rfloor$ to find remainder R.

R resulted out of each division is then compared with zero.

If R is non-zero for all cases, then M is prime.

When solution runs in MISD computer with PEs PE_2, PE_3, PE_5, $\cdots$, $\text{PE}_{\lfloor\sqrt{N}\rfloor}$ -

Each PE is loaded with M and divides M by potential divisor to get remainder R.

That is, PE_2 performs divide by 2, PE_3 performs divide by 3, PE_5 performs divide by 5, $\cdots$, $\text{PE}_{\lfloor\sqrt{N}\rfloor}$ performs divide by $\lfloor\sqrt{N}\rfloor$ simultaneously in O(1) time.

Here, M is the single data and division instructions form multiple instructions stream

Stream 2: divide M by 2, compare R with 0, ...

Stream 3: divide M by 3, compare R with 0, ...

Stream 5: divide M by 5, compare R with 0, ...

$\vdots$

Example 0.5 *Systolic array.*

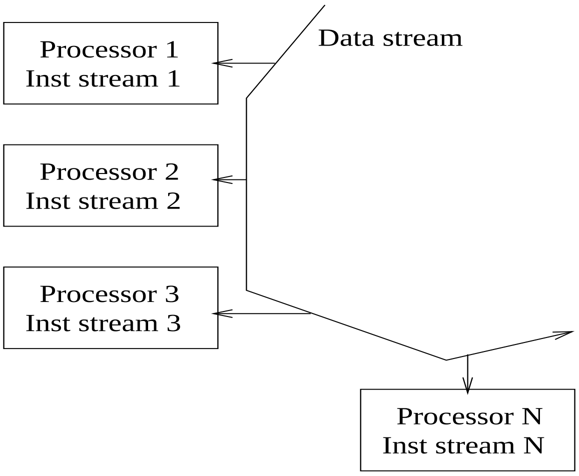


Figure 30: Systolic array

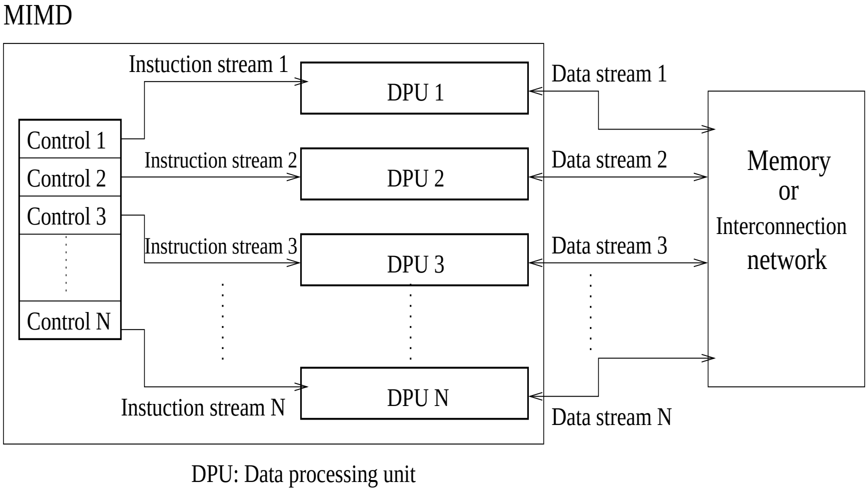


Figure 31: MIMD machine

0.5 MIMD computers

In MIMD m/c, each processor fetches its instruction and operates on its own data.

A processor of MIMD m/c is the general-purpose microprocessor.

Depending on interconnection architecture, MIMD machines are categorized as tightly coupled (shared memory) and loosely coupled (message passing).

SMP (symmetric multiprocessors) and NUMA (non-uniform memory access) can considered to be in tightly coupled MIMD.

Although in the current scenario, there is no such distinct boundary between tightly coupled and loosely coupled architectures.

0.5.1 UMA

In a shared memory system, processors exchange information through central shared memory. Each of the processors has the equal access right to memory and, therefore, called *balanced* as well as symmetric multiprocessor (SMP). This class of machine realizes UMA (uniform memory access).

Follow Figure 32. A processor has equal access time T for all memory locations.

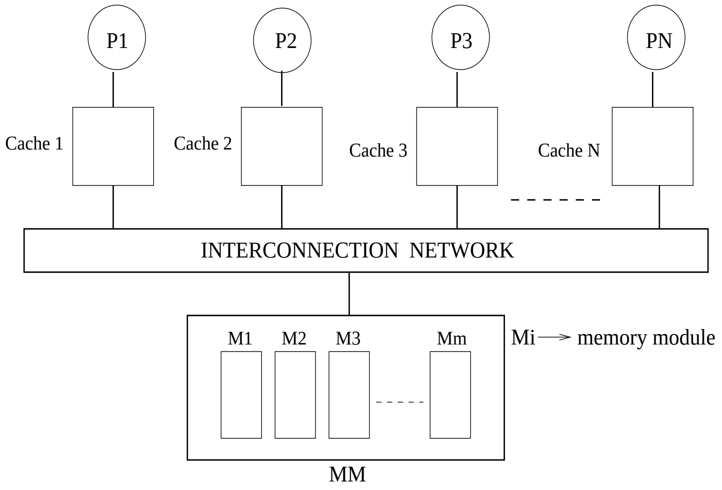


Figure 32: UMA architecture

The interconnection is commonly shared bus. However, the crossbar as well as multistage network are also in place.

0.5.2 NUMA

A loosely coupled MIMD computer is designed based on the physically distributed memory.

The physically dispersed memory is addressed by a logically shared address space.

Such a DSM (distributed shared memory or scalable shared memory) architecture is called the NUMA organization.

The access time to memory depends on location of memory.

Follow Figure 33. Each processor has a part of shared memory.

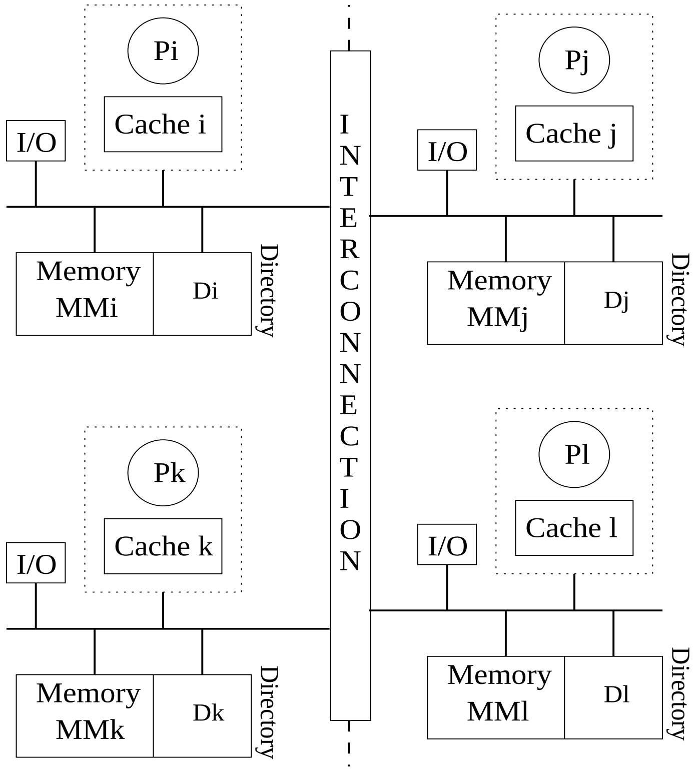


Figure 33: Non-uniform memory access

0.5.3 Cache only memory architecture

In cache only memory architecture (COMA), every memory block is a cache.

The data dynamically migrates close to a processor cache where it is most needed.

The structure of a COMA is shown in Figure 34.

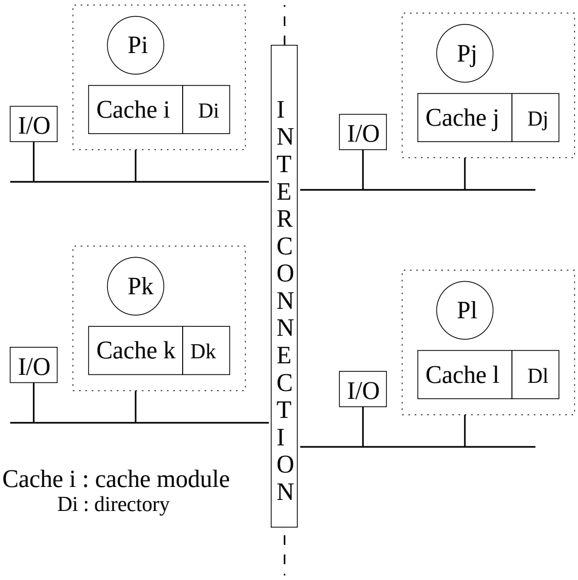


Figure 34: Structure of COMA machine

0.5.4 Multiprocessor system

Tightly coupled model of MIMD machine is called a multiprocessor system.

In such a system, processors communicate through shared main memory.

Each processor in such a system has fully programmable unit and can execute its own program. However, the set of processors form a single entity.

Data exchanges between processors are via a common bus system.

Figure 35 shows general structure of a multiprocessor system.

The interconnection network is the important part of a multiprocessor.

It is always designed according to the needs of application to be addressed.

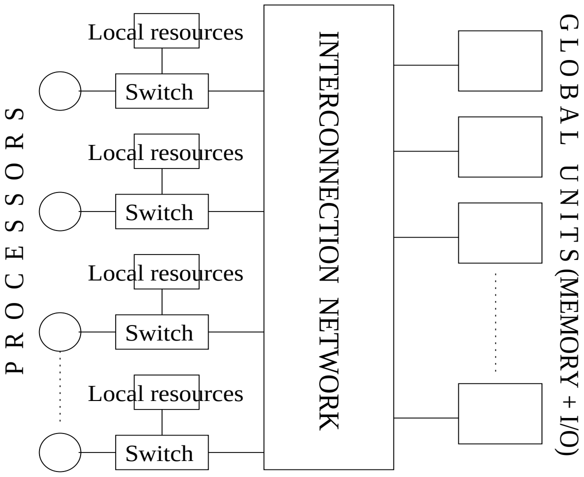


Figure 35: General structure of a multiprocessor system

Multiprocessor system uses common memory unit and shares common bus.

That is, two or more processors may request for same memory unit and, therefore, may need access to same communication link.

These issues are addressed through different effective interconnection network:

a. Common bus b. Multiport memory

c. Crossbar switches d. Crossbar multiple shared bus.

0.5.5 Shared bus

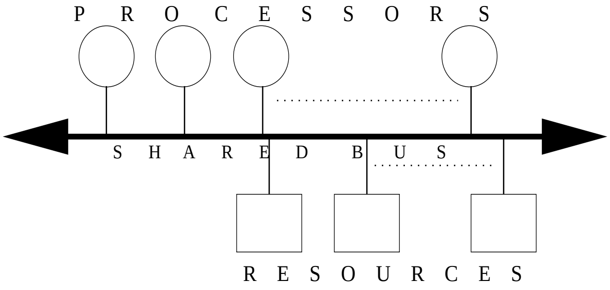


Figure 36: Multiprocessor shared bus

The shared bus architecture (also known as linear array).

Simple to implement.

Very much scalable. Addition and removal of units (processor/memory) are possible with marginal additional cost.

It allows transfer information at high speed and low cost.

A processor's information transferring unit checks whether bus is available. If bus is free, information can be sent to receiving unit.

However, shared bus architecture (Figure 36) is less reliable. Its reliability is as good as the reliability of weakest unit of the system.

Moreover, there can be a possibility of serious bottlenecks.

The shared bus architecture is limited by bandwidth of the bus and suitable only for a small system.

0.5.6 Multi-port

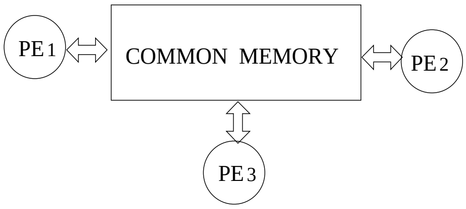


Figure 37: Multi-port common memory

Figure 37 is a three port (multi-port) memory architecture.

Content of memory (common memory of Figure 37) can be accessed simultaneously by the processing elements PE$_1$, PE$_2$, and PE$_3$ through three different ports.

In multi-port system, number of attached processors is restricted to number of ports.

0.5.7 Crossbar switch

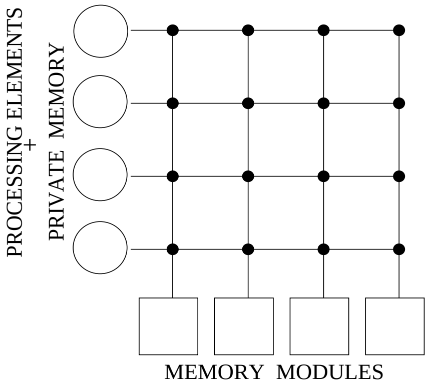


Figure 38: Crossbar switched multiprocessor system

In crossbar switched based multiprocessor system, a set of switches (dots in the figure) connects any processor to a memory module/unit.

It offers simultaneous access to all memory with least contention.

When same memory bank/module is requested by more than one processor at the same time then only bus contention may arise.

For an $n \times n$ crossbar at most n memory words can be delivered to at most n processors in each memory cycle.

That is, network complexity (measured in terms of number of switching points) is O(n^2). Thus such a network is suitable for small multi-processor system.

The crossbar switched multiprocessor systems is of high speed. A processor to memory path can be one crosspoint delay. However, a single stage crossbar network is not scalable once it is built.

0.5.8 Crossbar multiple shared bus

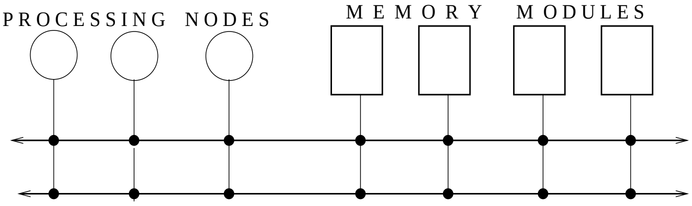


Figure 39: Crossbar multiple shared bus

It needs two sets of elementary switches (dots in Figure 39) to establish link between processors and buses and another sets to link between buses and memory modules.

Number of switches required in a system is

$(m+p)\times b,$

where m is the number of memory modules,

p is the number of processors and

b is the number of shared buses.

In Figure 39, m=4, n=3,and b=2.

0.6 Multistage networks

Multistage network uses $a \times b$ (a-input and b-output) elementary switches.

Such a design is less expensive than full crossbar switched based network.

Only incremental expansions are permitted in a network with elementary switches.

0.6.1 Elementary switch

Simple 2×2 elementary switch also is called interchange box (Figure 40(a)).

Its input A [B] can connect any one of the outputs 0, 1 depending on the value of control bit CONT$_A$ [CONT$_B$] supplied at A [B].

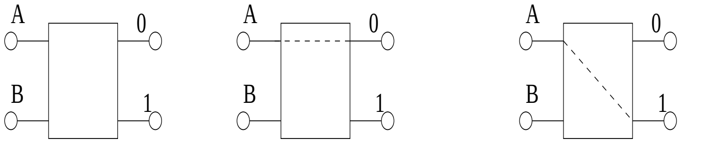
If CONT$_A$ [CONT$_B$] is 0, it connects the output 0 (Figure 40(b)).

To connect the output 1, the control bit is to be 1 (Figure 40(c)).

If both the inputs A and B demand connection to an output (say, 0), only one of those is allowed to connect to the output and the other one is blocked.

The general form of elementary switches is $a \times b$ -that is a-input and b-output.

For 2×2 switch both the a and b are 2.

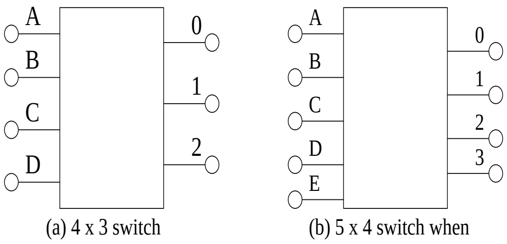


(a) 2 x 2 switch (b) 2 x 2 switch when CONT A= 0 (c) 2 x 2 switch when CONT B= 0

Figure 40: The 2×2 elementary switch

A control input CONT for a 4×3 (4-input, 3-output) switch can either be 0, 1, or 2 to select the output number 0, 1 or 2 (Figure 41(a)).

Similarly, for a 5×4 switch, the control bit CONT can take any one of the four values 0, 1, 2, and 3 for selection of its one of the four outputs (Figure 41(b)).



(a) 4 x 3 switch

(b) 5 x 4 switch when

Figure 41: The a×b elementary switches

QUIZ