Distributed Computing

Vectorization

Mahmoud Badry, mma18@fayoum.edu.eg

AGENDA

1 Flynn's taxonomy

2 Vectorization

3 Vectorization inhibitors

5 End

Questions

Flynn's taxonomy

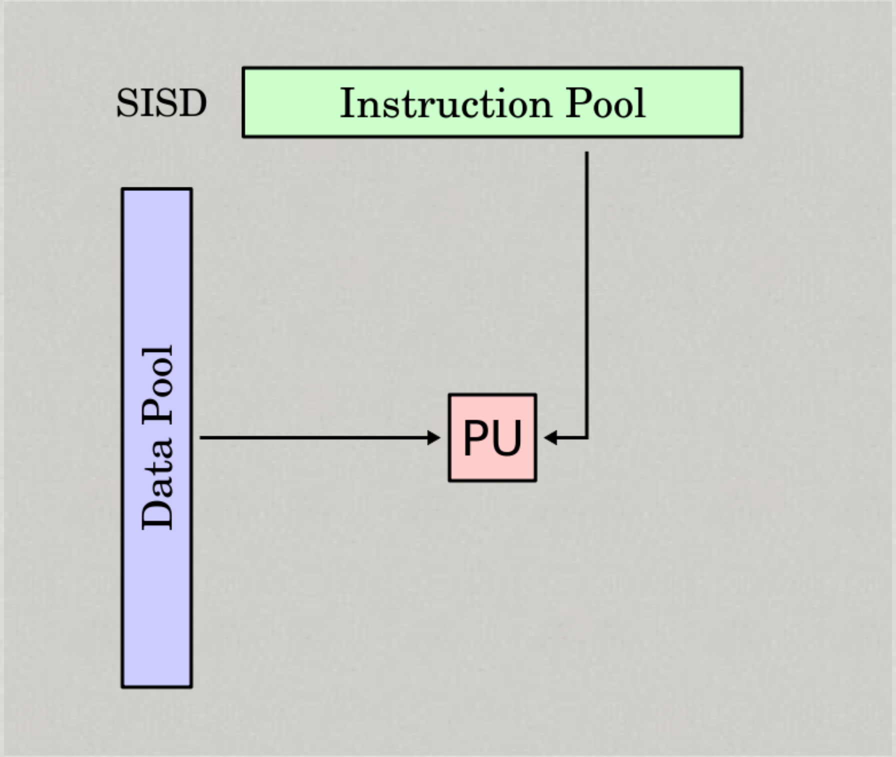
Computer Architectures

Mahmoud Badry, mma18@fayoum.edu.eg

Flynn's taxonomy

- Flynn's taxonomy is a **classification** of **computer architectures**, proposed by Michael J. Flynn in 1966.
- The **four** classifications defined by Flynn are based upon the number of concurrent instruction (or control) streams and data streams available in the architecture:
  - SISD
  - SIMD
  - MISD
  - MIMD

SISD (Single Instruction - Single Data)

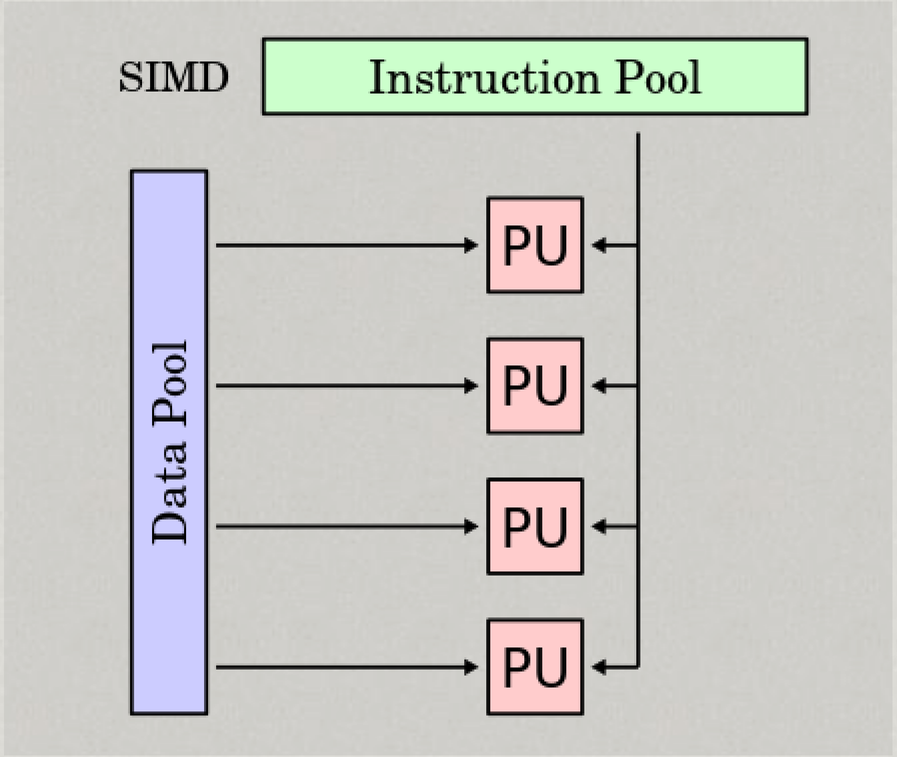


A sequential computer which exploits no parallelism in either the instruction or data streams. Single control unit (CU) fetches single instruction stream (IS) from memory. The CU then generates appropriate control signals to direct single processing element (PE) to operate on single data stream (DS) i.e., one operation at a time.

Examples:

personal computers (PCs; by 2010, many PCs had multiple cores)

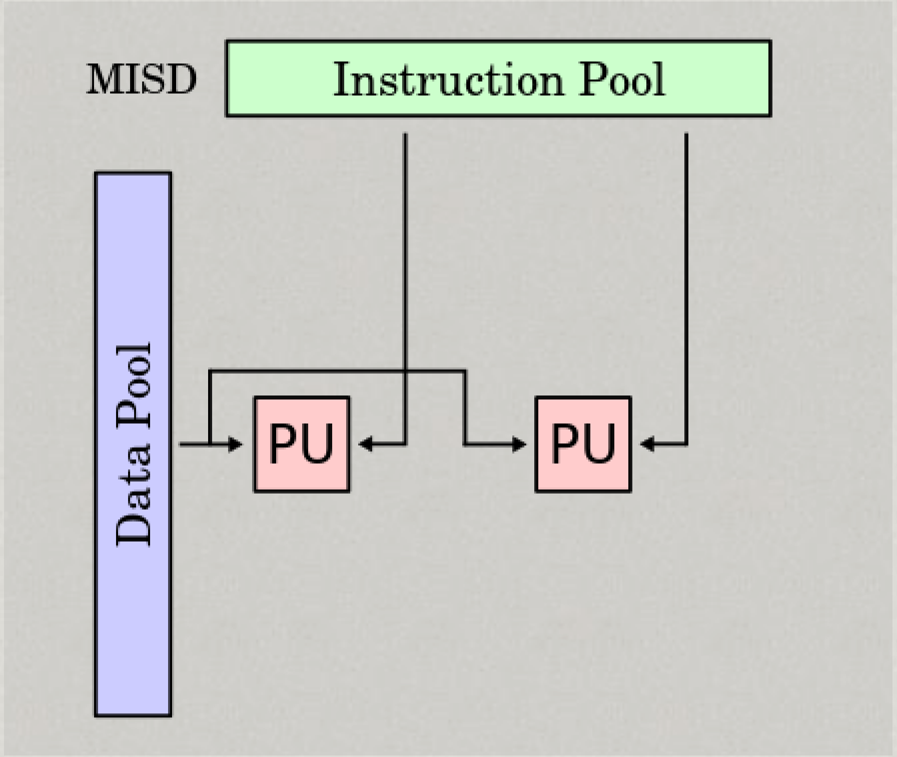
SIMD (Single Instruction - Multiple Data)



A computer which exploits multiple data streams against a single instruction stream to perform operations which may be naturally parallelized.

Examples:
Array processor or graphics processing unit (GPU)

MISD (Multiple Instruction – Single Data)



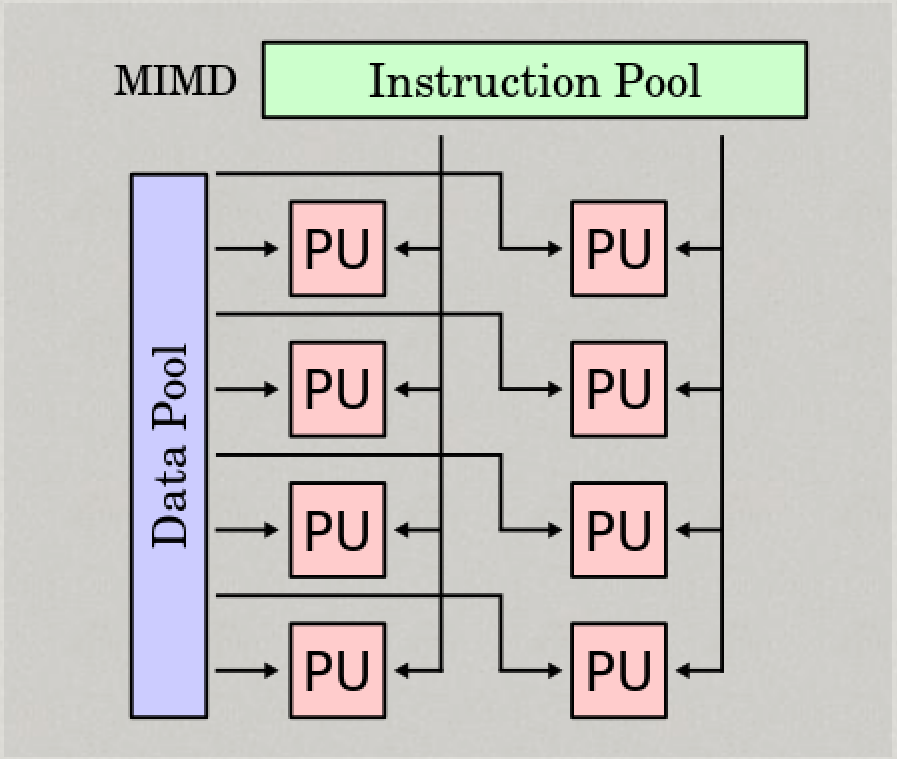
Multiple instructions operate on one data stream. Uncommon architecture which is generally used for fault tolerance. Heterogeneous systems operate on the same data stream and must agree on the result.

Examples:

The Space Shuttle flight control computer.

Mahmoud Badry, mma18@fayoum.edu.eg

MIMD (Multiple Instruction – Multiple Data)



Multiple autonomous processors simultaneously executing different instructions on different data.

Examples:

Multi-core superscalar processors (i.e. Intel i7)

Vectorization

What is Vectorization?

Mahmoud Badry, mma18@fayoum.edu.eg

Vectorization

- Vectorization is the process of rewriting a loop so that instead of processing a single element of an array N times, it processes (say) 4 elements of the array simultaneously N/4 times.
- Vectorization allows you to execute a single instruction on multiple data objects in parallel within a single CPU core, thus improving performance.

Vectorization Example

- Suppose you have this loop:

```
do i=1,100
    a(i) = b(i) + c(i)
end do
```

- The compiler first unroll the loop:

```
do i=1,100,4
  a(i)  = b(i) + c(i)
  a(i+1)= b(i+1) + c(i+1)
  a(i+2) = b(i+2) + c(i+2)
  a(i+3) = b(i+3) + c(i+3)
end do
```

- Then into:

```
do i=1,100,4
   <load b(i), b(i+1), b(i+2), b(i+3) into a vector register, vB>
   <load c(i), c(i+1), c(i+2), c(i+3) into a vector register, vC>
   vA = vB + vC
   <store a(i), a(i+1), a(i+2), a(i+3) from the vector register, vA>
end do
```

Automatic Vectorization

- Compilers can auto-vectorize loops for you that are considered safe for vectorization. In case of the Intel compiler, this happens when you compile at default optimization level (-O2) or higher. On the other hand, if you want to disable vectorization for any loop in a source file for any reason, you can do that by specifying the '-no-vec' compile flag.

Automatic Vectorization Visual C++

- To set the automatic vectorization compiler option in Visual Studio:
  - In Solution Explorer, open the shortcut menu for the project and then choose Properties.
  - In the Property Pages dialog box, under C/C++, select Command Line.
  - In the Additional Options box, **/O2 /Qvec-report:1 /Qvec-report:2**

E

Automatic vectorization exmaple

Mahmoud Badry, mma18@fayoum.edu.eg

Vectorization inhibitors

What stops loops from being vectorized?

Mahmoud Badry, mma18@fayoum.edu.eg

Trip Count changes

- The loop trip count must be known at entry to the loop at runtime.
- Statements that can change the trip count dynamically at runtime (such as Fortran's 'EXIT', computed 'IF', etc. or C/C++'s 'break') must not be present inside the loop.

Branching

- In general, branching in the loop inhibits vectorization. Thus, C/C++'s 'switch' statements are not allowed.
- However, 'if' statements are allowed as long as they can be implemented as masked assignments. The calculation is done for all 'if' branches but the results is stored only for those elements for which the mask evaluates to true.

Branching Example

```
for (int i=0; i<length; i++) {
        float s = b[i]*b[i] - 4*a[i]*c[i];
        if ( s >= 0 ) {
                s = sqrt(s) ;
                x2[i] = (-b[i]+s)/(2.*a[i]);
                x1[i] = (-b[i]-s)/(2.*a[i]);
        }
        else {
                x2[i] = 0.;
                x1[i] = 0.;
        }
}
```

Nested Loops

- Only the innermost loop is eligible for vectorization. If the compiler transforms an outer loop into an inner loop as a result of optimization, then the loop may be vectorized.

Function Calls

- A function call or I/O inside a loop prohibits vectorization.
- Intrinsic math functions such as 'cos()', 'sin()', etc. are allowed because such library functions are usually vectorized versions.
- A loop containing a function that is inlined by the compiler can be vectorized because there will be no more function call.

Data Dependency

- There should be no data dependency in the loop. Some dependency types are:
  - flow dependency: Read-after-write
  - anti-dependency: Write-after-read
  - output dependency: Write-after-write

Data Dependency: flow dependency

- Read after write example:

```
do i=2,n
   a(i) = a(i-1) + 1
end do
```

Data Dependency: anti-dependency

- Write-after-read example:

```
do i=2,n
   a(i-1) = a(i) + 1
end do
```

Data Dependency: output dependency

- Write-after-write example:

```
do i=2,n
  a(i-1) = x(i)
  ...
  a(i)   = 2. * i
end do
```

Non-contiguous memory access

- Non-contiguous memory access hampers vectorization efficiency.
- Eight consecutive ints or floats, or four consecutive doubles, may be loaded directly from memory in a single AVX instruction. But if they are not adjacent, they must be loaded separately using multiple instructions, which is considerably less efficient.

?

QUESTIONS

THANK YOU!