Program Efficiency Enhancement by Effective Caching

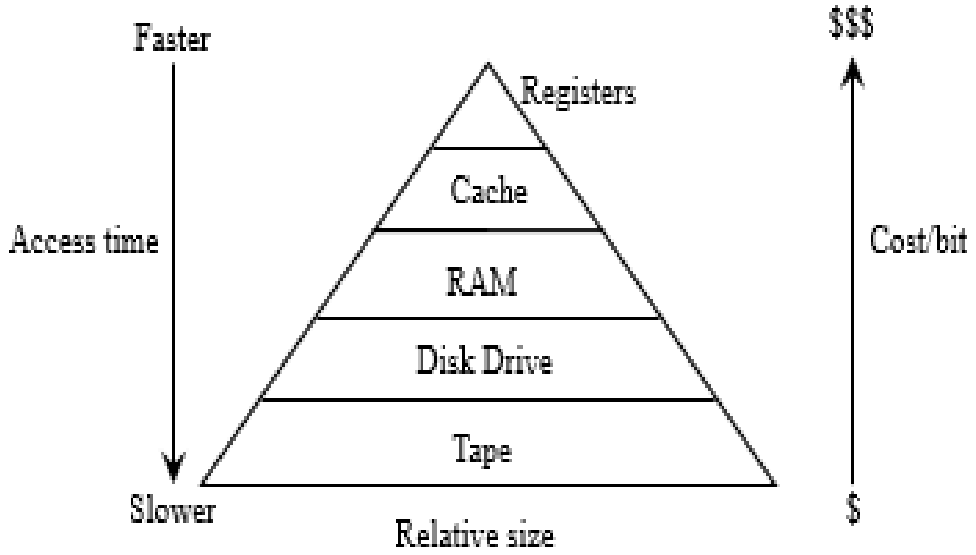
OVERVIEW OF THE **C** PROGRAMMING AND WRITING A BASH SCRIPT

Cache Memory

- An important part of the memory hierarchy in any computer system
- Physically located between the registers, which is part of the processor and random access memory (RAM)
- It is much larger than the number of registers and much smaller than the size of RAM
- Utilizes different technology compared to RAM
  - Faster
  - Uses more power
  - Costs more

Memory Hierarchy

- Gives the illusion of a large (RAM size), fast (cache speed) memory
- Compromise between cost, access time and size

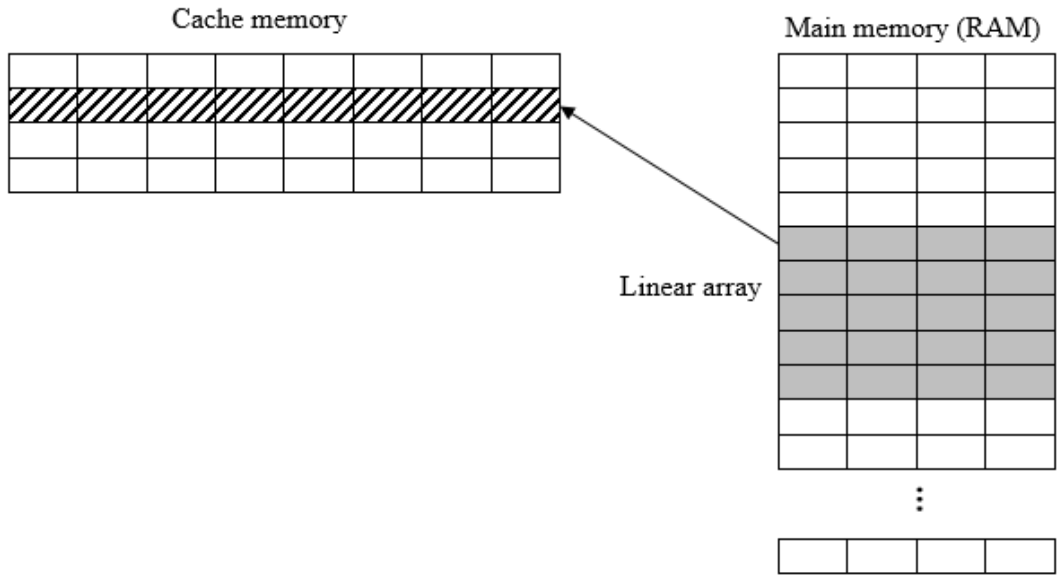


Cache Principles

- Principles of Locality
- Spatially
  - Likely nearby memory words will be accessed
  - More than one word is brought into the cache during a memory request
    - Cache line typically holds 64 bytes
- Temporally
  - Likely a memory word will be used again
  - Keep as much in the cache as possible (function of cache size)
  - Replace the least frequently used words when necessary

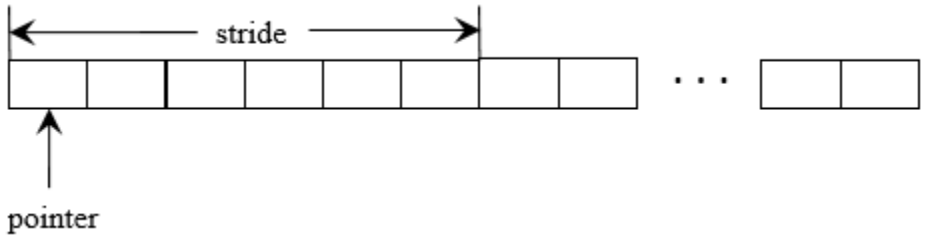
Efficient Memory Accesses

- Sequentially access each word in a linear array
  - First access word is not in the cache and RAM must be accessed (slow)
  - Some number of the next words are in cache (fast)
    - One slow access, n – 1 fast accesses



Program Overview

- The program name is **memStrideT**
- Allocates a linear array of some specified data type
- Initializes a pointer to the first element
- Iterates through all elements
  - It skips a specified number of elements called the **stride**
  - Makes multiple loops to ensure all elements are accesses
  - A value is written into each element to avoid compiler optimizations
- The time to access all array elements is output



C Language Primer

- The main function is a function that can be passed argumets
  - int main(int argc, char* argv[])
    - The argc argument means “argument count”
      - The number of arguments on the command line
    - The argv argument means “argument vector”
      - It is an array of pointers to characters (C string) representing the arguments value
    - If an argument is a number it needs to be converted from a **C** string into a decimal value, see the function strtol (string to long)

C Language Primer

- Dynamic memory allocation/deallocation
  - Use the **C** function named ***malloc***
  - It returns a pointer to a character, the first byte in the newly allocated memory block
  - The block is guaranteed to be contiguous memory locations
  - Use the **C** function ***free*** to give dynamic memory back to the Operating System
- Use the **C** function ***printf*** to output values

BASH Primer

- Linux shell can be programmed using a BASH script
- The first line in the ASCII script contains the complete pathname of the file (program) that should execute the file contents
  - **#!/bin/bash**
- Commands the would be typed can be put in the shell script
  - **make memStriderT**
- Shell scripts can define and use variables
  - **OUTFILE=memStrideDouble**
  - It is important there not be spaces

BASH Primer

- Shell scripts can contain conditional statements
- The following means remove the file if it exists in the same directory as the shell script

```
if [ -f $OUTFILE.csv ]; then
        rm -f $OUTFILE.csv
fi
```

- The value in a variable is accessed by preceding the variable name with a $ as in $OUTFILE above

BASH Primer

- Shell scripts can contain looping constructs

```
while [ $COUNTER -le 133 ]; do
        ./memStrideT 100 $COUNTER >> $OUTFILE.csv
        echo $COUNTER       #Progress Visualization
        let COUNTER=COUNTER+1
done
```

- This loop executes the program **memStrideT** with different values for the second argument, the number of elements to skip when accessing the linear array
- Program output is also appended to the output file during each iteration

Program Results

- The shell script takes approximately fifteen minutes to execute on a laptop with an i7 processor and 16 GB RAM
  - Surprisingly, the shell script takes even longer to execute on a massively parallel computer such as Blue Waters
- The program was executed with unsigned character, single- and double-precision floating-point elements
- In all cases, it appears the cache becomes inefficient when the stride is greater than 128 bytes
- To maximize cache efficiency it is best to access elements sequentially

Program Results