Main memory

• Programs are stored in persistent memory
• While a program is running, the program requires temporary memory to execute
• Long term memory can be slow, but memory required during execution must be relatively fast

Main memory

• To access main memory:
  – Each byte in main memory has a unique address
  – The CPU sends an address and either flags to either:
    • Retrieve the value of the byte at that address
    • Set the byte at that address to a specific value
Main memory

• Okay, so each byte has its own address
  – This is called byte-addressable

• If you want to change just one bit,
  you must use the bit-wise and
  bit-shift operators on a byte

Serial versus parallel communication

• If you read a 10 digit number to a friend,
  you are communicating serially; one digit at a time
  – Also, you must know when you're starting and stopping

• If you and nine friends each communicates one of those digits to one
  of ten corresponding friends,
  you are communicating in parallel; all ten digits at once

• The first is cheaper, the second is faster

• The communication between the CPU and main memory is parallel
  – A bus of n lines has each line carrying one bit of an address

Addresses

• In a computer,
  an address bus has n lines, each sending a 0 or a 1
  – This allows $2^n$ different addresses

• The Intel 386 was the first common CPU with a 32-bit address bus
  – 32 wires connected the CPU and main memory carrying the address

• The first common CPU with a 64-bit addresses was the Nintendo 64
  – 64 wires connected the CPU and main memory carrying the address

• Incidentally, the Commodore 64 had 64 KiB of main memory
  – 64 KiB = $2^{16}$ bytes
  – This could be addressed with a 16-bit address
Main memory

Addresses

- If every byte has its own address, then
  - A 32-bit address can uniquely address \(2^{32} = 4\) GiB
  - A 64-bit address can uniquely address \(2^{64} = 67,108,864\) TiB

- The restriction of 32-bit computers to accessing only 4 GiB of main memory led to the general adoption of 64-bit computers

Recall, however, that we can represent four bits with one hexadecimal digit

- By convention, we will
  - Leave off leading zeros
  - Use ellipsis for intermediate fs

- For example,
  \(\text{a310 instead of 0000a310}\)
  \(\text{f...fb08 instead of ffffffb08}\)

- If we are obviously discussing addresses, we may leave off the 0x

Thus, given this 32-bit address,
\(0b11101011011100010101011110\)
we could write it as
\(0xf56e155e\)

Similarly, given this 64-bit address in hexadecimal:
\(0x0003a56f29e5b8\)
we could determine the bits
\(0b00000000000110100110100001101101000111111010110110000000\)
Thus, we could visualize all of main memory as shown here.

- Assume this is a 32-bit computer with 4 GiB of main memory.

When a program is executed, the operating system allocates some block of memory for its execution.

For the purpose of this course, we will assume that the program has access to all of memory.

- This is actually achievable with virtual memory.

The instructions are stored starting at the top of memory.
- This is called the code segment.
• Literals are stored next in the data segment

• Memory for local variables is stored starting at the bottom of memory
  – This is called the call stack

• As we need more local variables, the call stack will grow towards the top of memory

• The remaining memory between the data segment and the call stack will be used for additional features:
  – Local variables that keep their value between function calls (static)
  – Dynamically allocated memory (the heap)

• Suppose we have this program:

```c++
#include <iostream>

int main()
{
    int data[5];
    std::cout << data[0] << std::endl;
    return 0;
}
```

Output:
```
0xffff3d80
```

```c++
```
Summary

- Following this lesson, you now
  - Know that main memory is byte addressable and each byte has its own unique address
  - Know addresses are passed in parallel through an address bus with a fixed number of $n$ lines or bits
  - Understand that this limits available main memory to $2^n$ bytes
  - Know that addresses are represented as hexadecimal digits
  - Understand that an executing program occupies a
    - Code segment
    - Data segment
    - Call stack
  - Are aware of how an array may be stored in main memory

References

[1] No references?

Colophon

These slides were prepared using the Georgia typeface. Mathematical equations use Times New Roman, and source code is presented using Consolas.

The photographs of lilacs in bloom appearing on the title slide and accenting the top of each other slide were taken at the Royal Botanical Gardens on May 27, 2018 by Douglas Wilhelm Harder. Please see https://www.rbg.ca/ for more information.

Disclaimer

These slides are provided for the ECE 150 Fundamentals of Programming course taught at the University of Waterloo. The material in it reflects the authors’ best judgment in light of the information available to them at the time of preparation. Any reliance on these course slides by any party for any other purpose are the responsibility of such parties. The authors accept no responsibility for damages, if any, suffered by any party as a result of decisions made or actions based on these course slides for any other purpose than that for which it was intended.