CSCI 8530
Advanced Operating Systems

Hardware and Xinu

Instructor: Pei-Chi Huang

Review: Goal

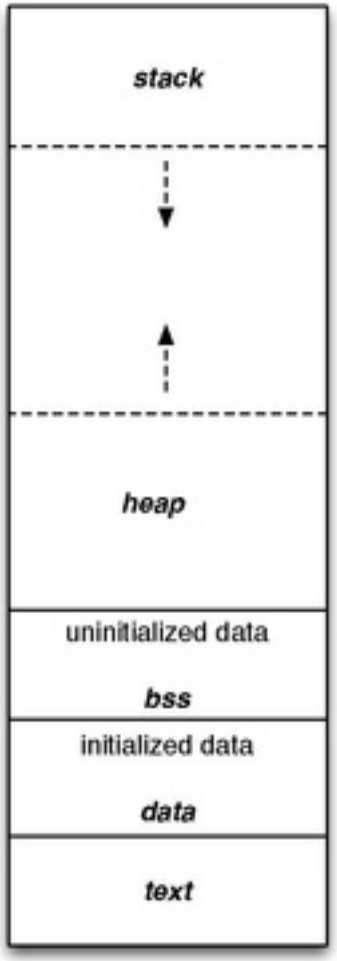
- We'll be studying a real operating system at the code level to gain an understanding of what's really involved in an operating system.
  - You can't debug using traditional methods because you're writing at a lower level.

Review: Prerequisites (1/2)

- C or C++ Programming experience
  - Optional: Know pointers, structs and the basic data types, control loops, and the preprocessor well
- Know the assembly language and the architecture of the machine
- Need at least one machine that can be rebooted frequently for testing
  - Maybe another machine for editing and/or compiling
  - Intel Galileo (Gen 2) boards & Linux
- Persistence and Mental Stability
- Concepts from a previous OS course

Review: Prerequisites (2/2)

- Understanding of runtime storage components
  - Segments (text, data, and bss) and their layout
  - Runtime stack, activation records, and argument passing
  - Basic heap storage management (free list)
- Working knowledge of basic UNIX tools
  - Text editor (e.g., vi)
  - Compiler / linker / loader
  - Tar archives
  - Make and Makefiles



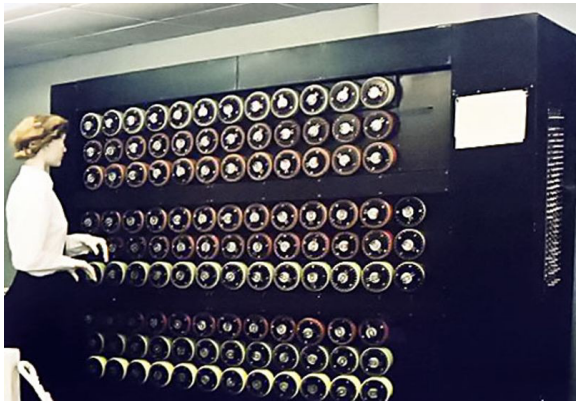
Review: the Answers for Last Time

1. What is the pwd command?
   - print working directory command.
2. How do you insert comments in the command line prompt?
   - #
3. Write a command that will display all .txt files.
   - Ls –al *.txt
4. How can you append one file to another in Linux?
   - cat file2 >> file 1
5. Explain how you can find a file using Terminal?
   - find . –name “filename.txt”
6. Explain how you can view the text file using Terminal?
   - cd
7. How do you change permissions under Linux?
   - Chmod
8. How to create symbolic link/hard link
   - ln -s

How Did We Get Here?

- 1940s – 1950s: How can computers be built?
  - The dawn of digital computers
  - Each processor is special-purpose
  - Each I/O device has a unique interface
  - Software is written in assembly language
  - Programs are written to control the hardware

The first Bombe is completed



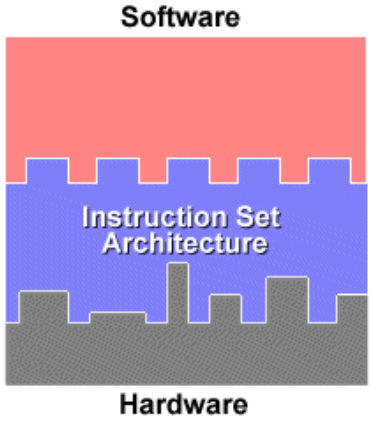
Bombe replica, Bletchley Park, UK

First Computer Program to Run on a Computer

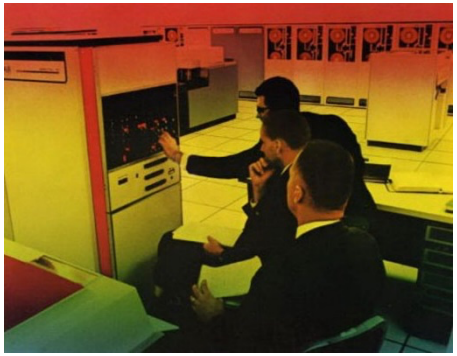
Small-Scale Experimental Machine (SSEM), 1941

How Did We Get Here?
(continued)

- 1960s: What are the best abstractions to use?
  - A move to general-purpose hardware
  - *Instruction Set Architecture* emerges
  - Families of computers devised
  - I/O became independent of specific hardware details
  - High-level programming languages created (e.g., parameterized functions, data types, and recursion)

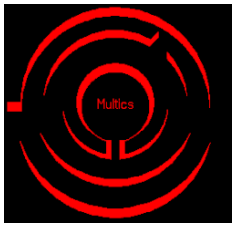


The first large commercial computers to use integrated circuits



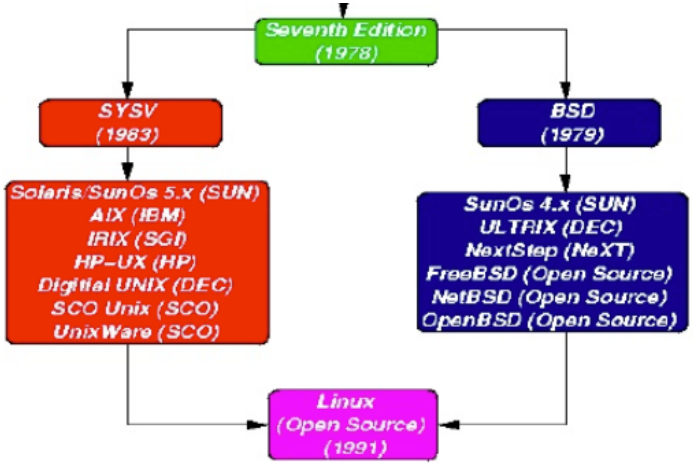
Spectra systems can run IBM 360 instruction set, 1968

How Did We Get Here? (continued)

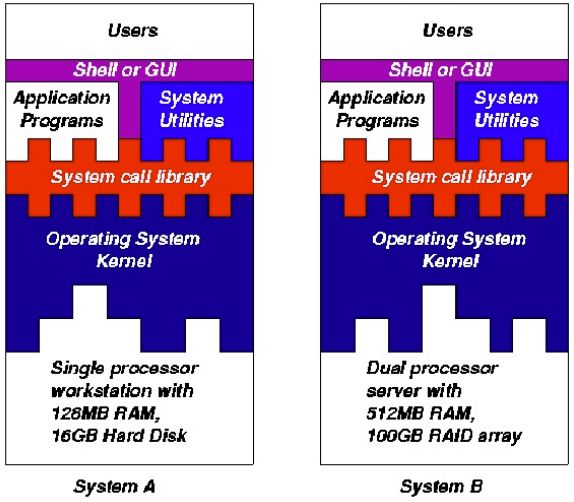


MIT, GE, Bell Labs, 1969

- Late 1960s and 1970s
  - Researchers investigate operating systems
  - The Multics project devises brillant abstractions
    - Multiplexed Information and Computing Service is an influential early time-sharing OS, based around the concept of a single-level memory
  - Unix introduces simple, elegant, and efficient versions of the Multics abstractions
- 1980s and on
  - The Internet project discovers a set of communication abstractions that are elegant and efficient
  - All systems are connected



Simplified UNIX FamilyTree



General OS Architecture

What Can We Conclude From History?

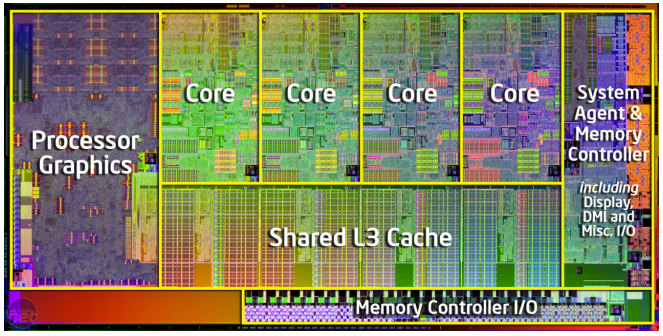
- The gap between hardware and users is huge.
  - Hardware is ugly.
  - Operating system abstractions are beautiful.
- Everything is connected.
  - Data centers
  - Desktops
  - Smart phones
  - Devices
  - Sensors

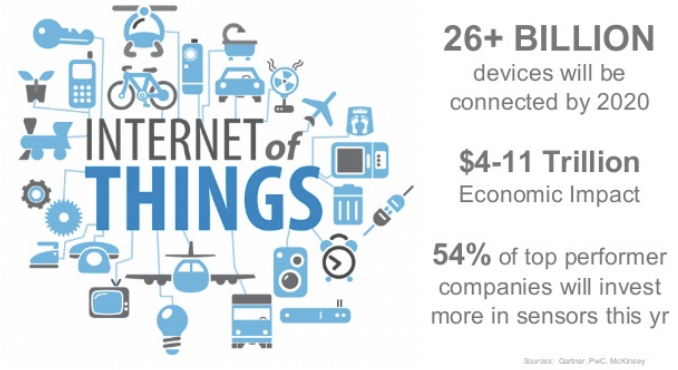
Consequences and Observations

- Our job in Computer Science is to build beautiful new abstractions that fill the gap between ugly hardware and users.
- We cannot consider operating systems without including computer networking.
- The central questions in computing have always focused on the tradeoff between imagined beauty and performance.
  - It is easy to imagine new abstractions.
  - We must restrict ourselves to abstractions that map onto the underlying hardware efficiently (and hope that hardware engineers eventually adapt to our abstractions).

The Once and Future "Hot Topic"

- In the 1970s and early 1980s, operating systems was one of the hottest topics in CS
- By the mid-1990s, OS research had stagnated
- Now things have heated up again, and new operating systems are being designed for
  - Multicore systems
  - Data centers
  - Smart phones
  - Large and small embedded devices
  - The Internet of Things





The xinu operating system

Motivation for Using a Real System

- Provides examples of the principles
- Makes everything concrete
- Gives students a chance to experiment
- Shows how abstractions map to real hardware

Why Xinu?

- Small — can be read and understood in a semester
- Complete — includes all the major components
- Elegant — provides an excellent example of clean design
- Powerful — has dynamic process creation, dynamic memory management, and flexible I/O
- Practical — has been used in real products

What is Xinu?

- Unix spelled backward
  - Although it shares concepts and even names with Unix, the internal design differs completely
- Developed by Dr. Douglas E. Comer (Purdue University) University in the 1980s
  - The Xinu lab located in the Computer Science Department at Purdue University
- ANSI-compliant C
- Meant for educational purposes
  - A set of front-end machines (standard workstations running Linux)
  - A set of back-end machines (machines that are only used to download and test code)

Experimenting With Xinu

- Xinu available for platforms
  - x86, ARM, and MIPS.
- The second edition of the text contains code for both the Intel x86 (Galileo) and ARM Cortex 8 (BeagleBone Black)
- Galileo has two ways to get started
  - Using an SD memory card and a serial cable.
  - Configure the board to boot over the network, and configure the PC to run DHCP and TFTP server processes
    - Recompile and download quickly once the pieces are in place

https://xinu.cs.purdue.edu/#code

https://xinu.cs.purdue.edu/files/Xinu_Galileo_Manual_v2.pdf

Some Technical Terms

- Cross Compiler
- Shell
- Kernel

Cross Compiler

- Compile a program on machine A
- Run it on machine B where an OS is not supported

Shell

- User interface for the operating system
- Access to the kernel
- Other programs are called through shell
- CLI (Command Line Interface)
- GUI (Graphical User Interface) – not available in our system

Kernel

- Lowest level but the most important part of an operating system
- Resource management
- Bridge between software and hardware

How We Will Proceed

- We will examine the major components of an operating system
- For a given component we will
  - Outline the functionality needed
  - Learn the key principles involved
  - Understand one particular design choice in depth
  - Consider implementation details and the relationship to hardware
  - Discuss other possibilities and tradeoffs
- Note: we will cover components in a linear order that allows us to understand one component at a time without relying on later components.

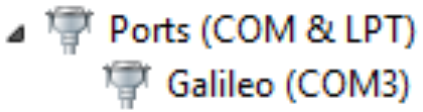
Intel Galileo Gen 2

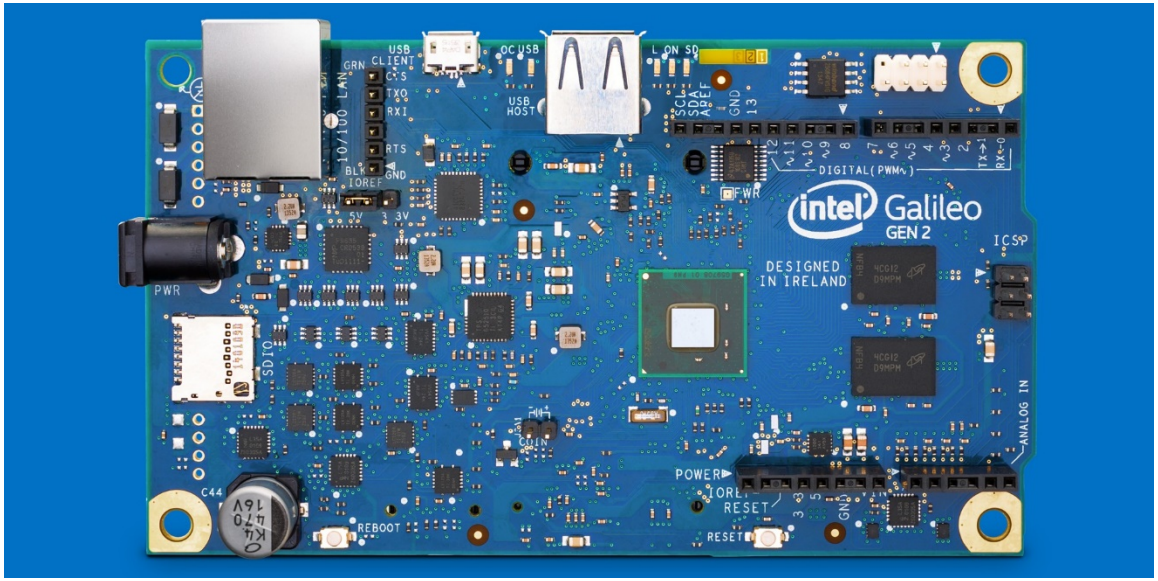
Intel Galileo Board

- Platform for prototyping embedded systems, sensors, robotics, electronics hacking, Internet of Things, no soldering
- $60 retail, Gen 2 board; latest version is Edison with WiFi/Bluetooth
- Compatible with Arduino Uno shields (expansion boards)
- Quark SoC which is a 32 bit single core single threaded Pentium CPU at 400 Mhz
- Can program with Arduino “sketches” or node.js, python
  - Not with ours unless put linux image on microSD card
- https://software.intel.com/iot/getting-started

Caveats

- Not 100% Arduino compatible (drivers, libraries)
- No wifi module for us
- No SD card
  - **Programs go away** when unplugged
  - Can't use Intel IoT or Eclipse IDE, must use Arduino IDE
- You have to install on your own machine
  - Install drivers then Arduino IDE
- Driver setup
  - Check your COM port, it may change when you reconnect




intel Galileo
GEN 2
DESIGNED
IN IRELAND
PWR
SDIO
REBOOT
RESET
POWER
DIGITAL(PWM~)
ANALOG IN
ICSP
USB
CLIENT
USB
HOST
OC USB
LAN
10/100
IOREF
5V
3.3V
GND

Galileo

Introduction

- Features
  - Intel® Quark™ SoC X1000 application processor
    - a 32-bit, single-core, single-thread, Intel® Pentium® processor instruction set architecture (ISA)-compatible
    - operating at speeds up to 400 MHz.

https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/galileo-g2-datasheet.pdf

Introduction

- Features
  - Support for a wide range of industry standard I/O interfaces,
    - including a full-sized mini-PCI Express slot
    - 100 Mb Ethernet port
    - microSD slot
    - USB host port
    - USB client port.

https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/galileo-g2-datasheet.pdf

Introduction

- Features
  - 256 MB DDR3
  - 512 kb embedded SRAM
  - 8 MB NOR Flash
  - 8 kb EEPROM standard on the board
  - support for microSD card up to 32 GB.

https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/galileo-g2-datasheet.pdf

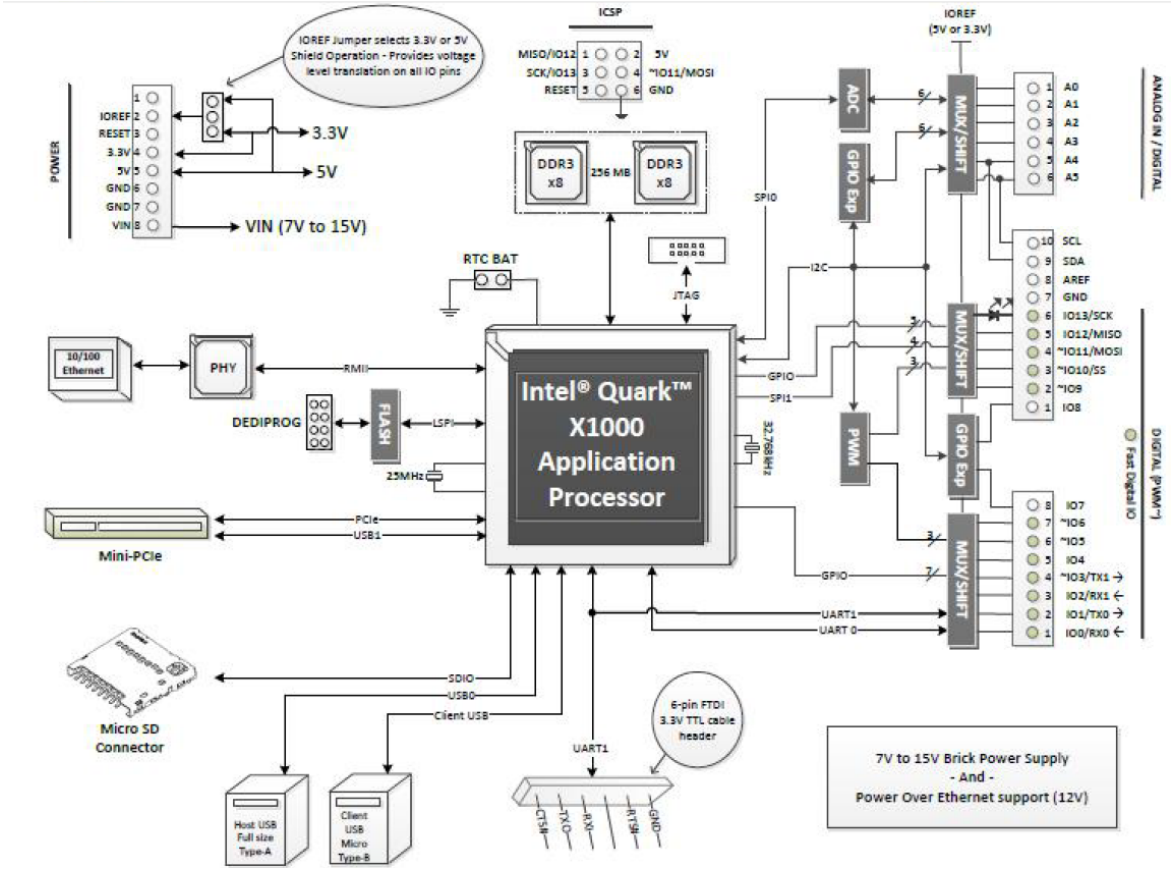
Introduction

- Features
  - Hardware and pin compatibility with a wide range of Arduino Uno R3 shields
  - Programmable through the Arduino integrated development environment (IDE) that is supported on Microsoft Windows, Mac OS, and Linux host operating systems.
  - Support for Yocto 1.4 Poky Linux release.

https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/galileo-g2-datasheet.pdf

Introduction

- Block Diagram

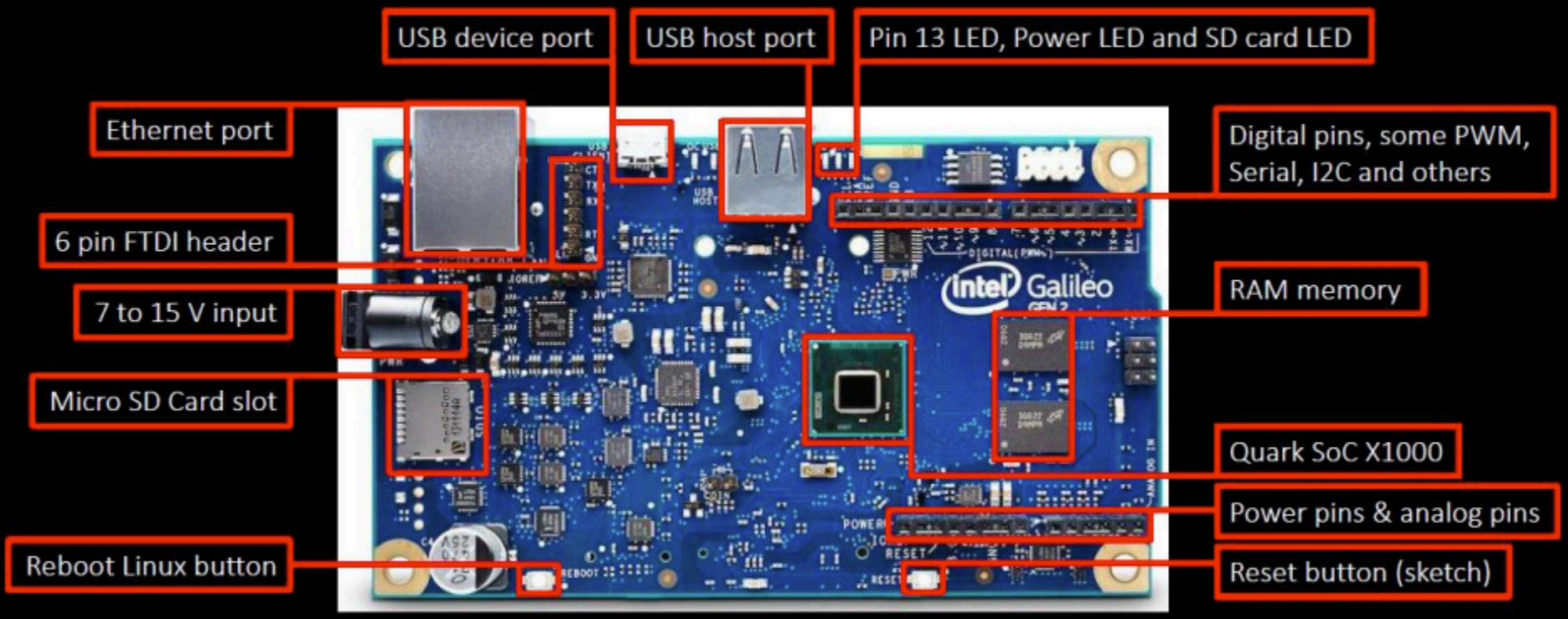


https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/galileo-g2-datasheet.pdf

Introduction

- DDR3: It is known as double data rate type 3 synchronous dynamic random access memory, with high bandwidth interface.(DRAM)
  - RAMS, are generally work as an intermediator between software location and processor,
  - In general, code is installed on some location whose position is loaded on RAM, which is accessed by processor.
  - Then after, instruction from processor, RAM loads the program on its memory interface for desired time,
- SRAM: It is known as static random access memory, which is faster than DRAM, and is generally used to store data but loses data as soon as power is off, but in galileo, it acts like a cache memory.
- PCIe: Peripheral Component Interconnect expressway.
  - As it provide lower latency and higher data transfer than parallel buses(Dedicated point to point connection)
- Clock: The CPU requires a fixed number of clock ticks (or *clock cycles*) to execute each instruction. The faster the clock, the more instructions the CPU can execute per second.
  - It is generally termed as clock rate, or speed at which the processor executes instruction.

Intel® Galileo (Gen 2)



https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/galileo-g2-datasheet.pdf

Intel® Galileo (Gen 2) – Software

- Supported IDEs
  - Arduino*
  - Eclipse*
  - Intel® XDK IoT Edition
    - C/C++
    - JavaScript*
    - Node.js*
    - Python*
- Supported OS
  - Yocto
  - Windows 8*
  - FreeRTOS

Intel® Galileo – Useful links

- Galileo downloads:
  - https://software.intel.com/en-us/iot/hardware/galileo/downloads
- Yocto Project – Default Linux distro's build system
  - https://www.yoctoproject.org/tools-resources/projects/poky
- Videos Competition - Intel Embedded Systems every year

Warning

- The input voltage to the Galileo board when it's using an external power source (as opposed to the supply connected at the power jack).
- You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- Warning: The voltage applied to this pin must be a ***regulated*** 5V supply otherwise it will damage the Galileo board.

Warning

- ***Warning:*** You must use a power supply to power the board! **You will damage the board if you power it with a USB cable.**
  - Connect the 12 V power cable to the Intel® Galileo Gen 2 board and to a power outlet.
  - Always connect the power **before** any other connection. The Power LED shown above (board label = ON) will turn on.

FINALLY, A FEW THINGS TO CONSIDER…

Perfection [in design] is achieved not when there is nothing to add, but rather when there is nothing more to take away.

– *Antoine de Saint-Exupery*

A teacher's job is to make the agony of decision making so intense you can only escape by thinking.

– source unknown

Real concurrency — in which one program actually continues to function while you call up and use another — is more amazing but of small use to the average person. How many programs do you have that take more than a few seconds to perform any task?

(From an article about new operating systems for the IBM PC in the New York Times, 25 April 1989)

Anything sufficiently weird must be fishy.

(The "ultimate rule" as suggested by Captain Shi Qiang in The Three-Body Problem *by Cixin Liu)*

References

- https://learn.sparkfun.com/tutorials/galileo-getting-started-guide?_ga=1.175151777.2095379488.1416564981
- http://akizukidenshi.com/download/ds/intel/Galileo_GettingStarted_329685_007.pdf