Platform-Based Development: Overview of Platforms

BS UNI studies, Spring 2024/2025

Dr Veljko Pejović
Veljko.Pejovic@fri.uni-lj.si

FRI
UNIVERSITY OF LJUBLJANA
Faculty of Computer and Information Science

What is a Platform?

- Evolution of computing
  - A single computer used by a few people for scientific computations
  - One computer per person for a range of tasks
    - Text editing, games, spreadsheets
  - Many computers per person
    - Embedded in everyday objects for specific tasks

What is a Platform?

- Evolution of computing
  - Individual “branches” are constantly evolving
  - Computers are networked
  - Needs diversified
    - AI chatbots
    - Image processing
    - Communication
    - Gaming
    - Sensing and actuating
    - Security
    - … and many others

What is a Platform?

- Different environments emerged to support application development in such a diverse landscape of needs
- Environment = hardware and/or software
  - Computer architectures
  - Operating systems
  - Application programming interfaces (APIs)
  - Programming frameworks

This is a very loose definition!

Platform is the environment in which a software application is executed

Platform – Abstraction Levels

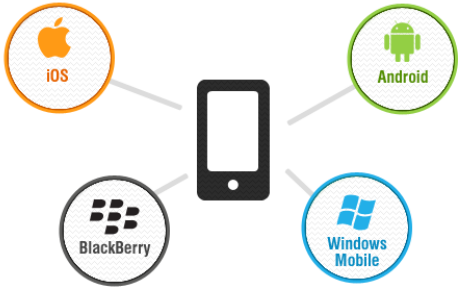
- From “32-bit platform” to “Platform-as-a-Service”
- Hardware level:
  - Processing capabilities (clock frequency, word length, memory size, instruction set)
  - Input/output capabilities (monitor, sensors, actuators)
- Middleware level:
  - Drivers, Operating system
- Higher level:
  - APIs, Cloud computing support
- What is the highest level?

We won’t consider software as a service (SaaS), e.g. Dropbox, Google Apps to be platforms

Others might consider them!

Platforms – (Incomplete) Breakdown

- Web platforms
- Mobile platforms
- Industrial/Embedded platforms
- Gaming platforms



Web Platforms

Web Platforms

Read my 1996 article for lab session discussion!

- The Web ≠ The Internet
  - However, The Web is the most popular service running on top of the Internet, thus for many of the 4 bn Internet users these two are synonyms
- In 1989 Tim Berners-Lee joined hypertext with the Internet to create the (World Wide) Web
- Hypertext: text with references that can be immediately accessed

Hypertext Idea

- Vannevar Bush in 1945:

“When data of any sort are placed in storage, they are filed alphabetically or numerically, and information is found (when it is) by tracing it down from subclass to subclass. It can be in only one place, unless duplicates are used; one has to have rules as to which path will locate it, and the rules are cumbersome. Having found one item, moreover, one has to emerge from the system and re-enter on a new path.

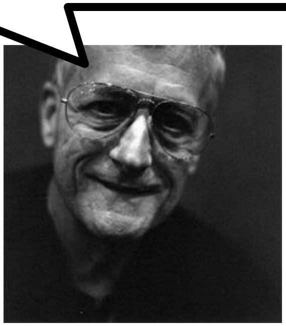
The human mind does not work that way. It operates by association. With one item in its grasp, it snaps instantly to the next that is suggested by the association of thoughts, in accordance with some intricate web of trails carried by the cells of the brain.“

Read my 1945 article to see how I predicted Wikipedia, speech recognition, even Google Glass!

Filing cabinet

Web of associations

Hypertext Idea



I meant hyper as extension and generality, not mere linking!

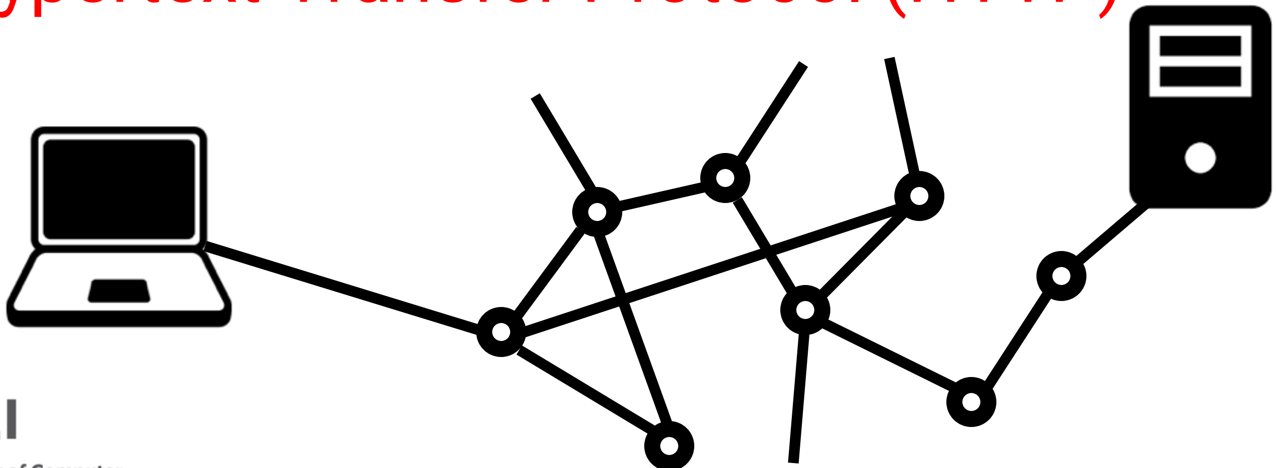
- 1963 – Ted Nelson coins the words hypertext and hypermedia
  - Also presents file system ideas for its implementation
- 1968 – Douglas Engelbart demonstrates NLS a system which implements hypertext linking
  - "The Mother of All Demos": also demonstrated the computer mouse, windows, video-conferencing, collaborative document editing, etc.
- Project Xanadu
  - A vision of "digital repository scheme for world-wide electronic publishing" - never implemented (properly)

Towards The Web Platform

- Hypertext implementations still failed to reach the visions
- It became clear that we need:
  - Information storage
    - Always available
    - Distributed
  - Connectivity
    - Worldwide
    - With (content) addressing and routing

Tim Berners-Lee’s WWW

- WWW creates a global information space
- Storage implemented via Web servers
- Hypertext documents written in HTML
- Communication over the Internet with a specific client-server protocol for content linking – Hypertext Transfer Protocol (HTTP)



HTTP’s Properties

- Request-response client-server protocol
  - Web browser is a client – issues requests
  - Web server is a server – sends responses
  - Example requests: GET, POST, PUT, DELETE
- Uniform resource locators (URLs) for resource referencing
  - In practice translates to a server’s IP address
  - Domain Name System (DNS) performs the translation
- Stateless protocol
  - Some Web applications use Cookies
- HTTPS for encrypted communication

HTTP Request and Response

- GET request

```
GET /Veljko/index.html HTTP/1.1
```

Path to the requested resource

Protocol version

```
HTTP/1.1 200 OK
```

If resource present and access granted

```
HTTP/1.1 404 Not found
```

If not found (diff. response codes for not authorised, etc.)

HTTP Request and Response

- POST request

```
POST /Veljko/form.html HTTP/1.1

 firstname = Veljko; lastname = Pejovic;
```

POST parameters in the body

```
HTTP/1.1 200 OK
```

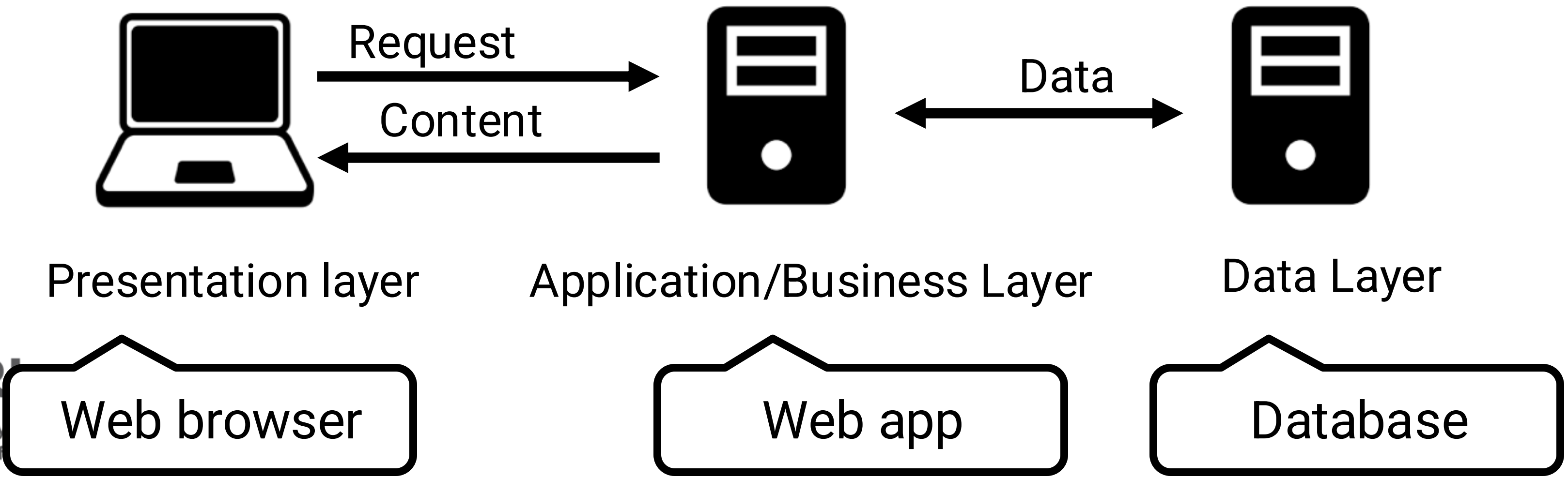
If request posted successfully

```
HTTP/1.1 405 Method
         not allowed
```

If not supported by the referred script/server

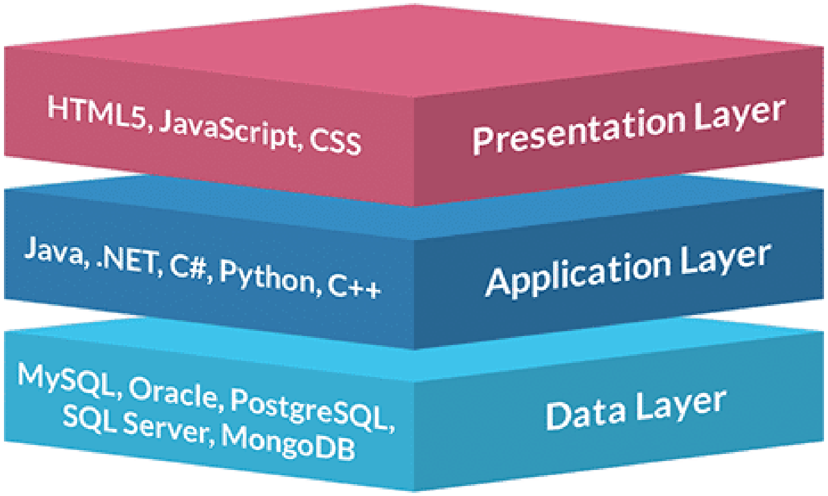
Web Platform Evolution

- Single-tier (AKA one-tier) architecture:
  - User interface, business logic, and data all within the same programme
- Multi-tier architecture:
  - Different layers for presentation, logic, data
  - More flexible, allows independent scaling and upgrading of different parts of the application



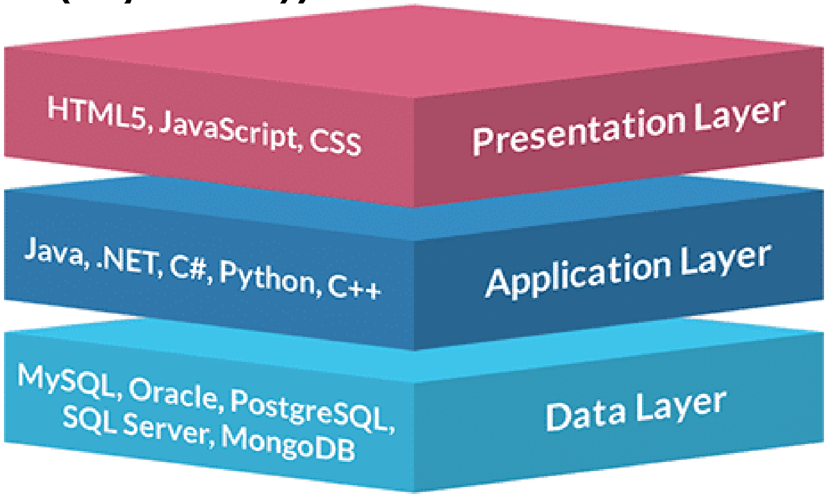
Implementing Multitier Web Platform

- Each tier with its own programming environment and language
- Presentation layer (AKA frontend): dealing with UI, thus focus on visual appearance, interaction
  - – Initially HTML, CSS
  - – Then JavaScript
  - – AJAX - **A**synchronous **J**avaScript **A**nd **X**ML
    - jQuery
    - Fetch
  - – Frameworks: Angular, Bootstrap, React, etc.


HTML5, JavaScript, CSS
Presentation Layer
Java, .NET, C#, Python, C++
Application Layer
MySQL, Oracle, PostgreSQL, SQL Server, MongoDB
Data Layer

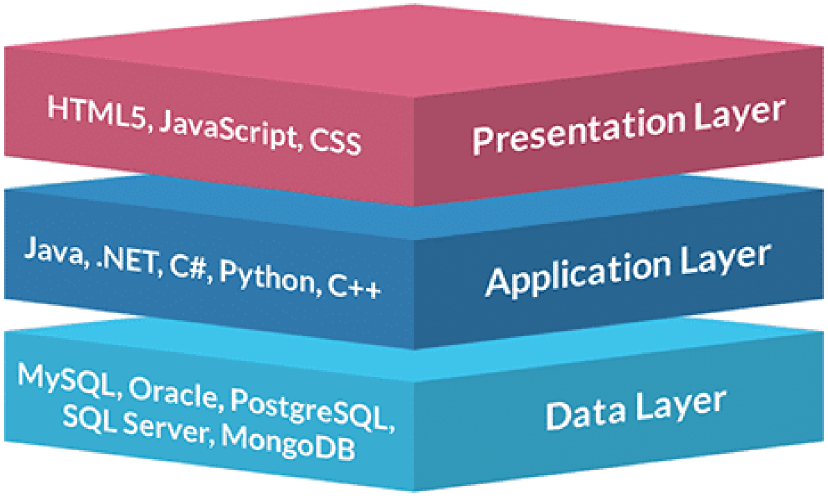
Implementing Multitier Web Platform

- Application/Business layer (AKA backend): dealing with app logic, data fetching and processing, interacting with the frontend
  - Web servers: Apache, nginx, IIS
  - Frameworks: Flask, Django (Python), JavaServer Pages-JSP, Drupal (PHP), ASP.NET,
  - Popularity of JavaScript frameworks, e.g. Node.js, Express.js


HTML5, JavaScript, CSS
Presentation Layer
Java, .NET, C#, Python, C++
Application Layer
MySQL, Oracle, PostgreSQL, SQL Server, MongoDB
Data Layer

Implementing Multitier Web Platform

- Data layer: dealing with data storage and organisation
  - Relational databases: data is stored in tables; relationships among data elements; theoretical underpinnings, ACID
    - MySQL, PostgreSQL
  - Non-relational databases: data is stored in different ways: key-value pairs, column stores, docs., etc.
    - MongoDB, Cassandra Influx DB


HTML5, JavaScript, CSS
Presentation Layer
Java, .NET, C#, Python, C++
Application Layer
MySQL, Oracle, PostgreSQL, SQL Server, MongoDB
Data Layer

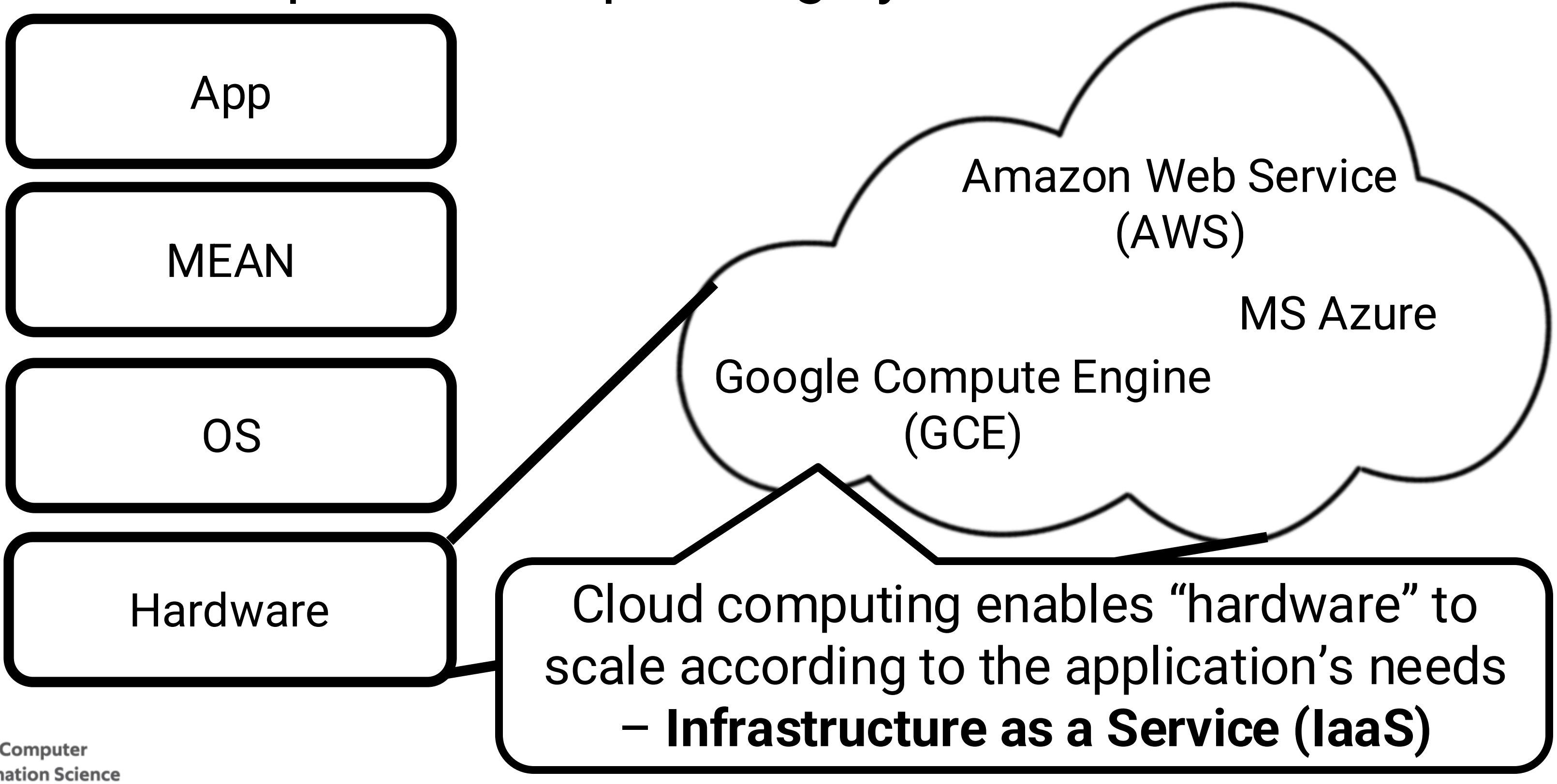
Web Platform – Current Standard

- MEAN architecture
  - MongoDB – for data storage
  - Express.js – as a Web server framework
  - Angular – for frontend development
  - Node.js – for event-driven server-side and networking
- What makes the above combination popular?
- What about the underlying technology (e.g. OS)?

A single language for both frontend and backend (JavaScript). In addition, JavaScript Object Notation (JSON) storage!

Web Platform – Current Standard

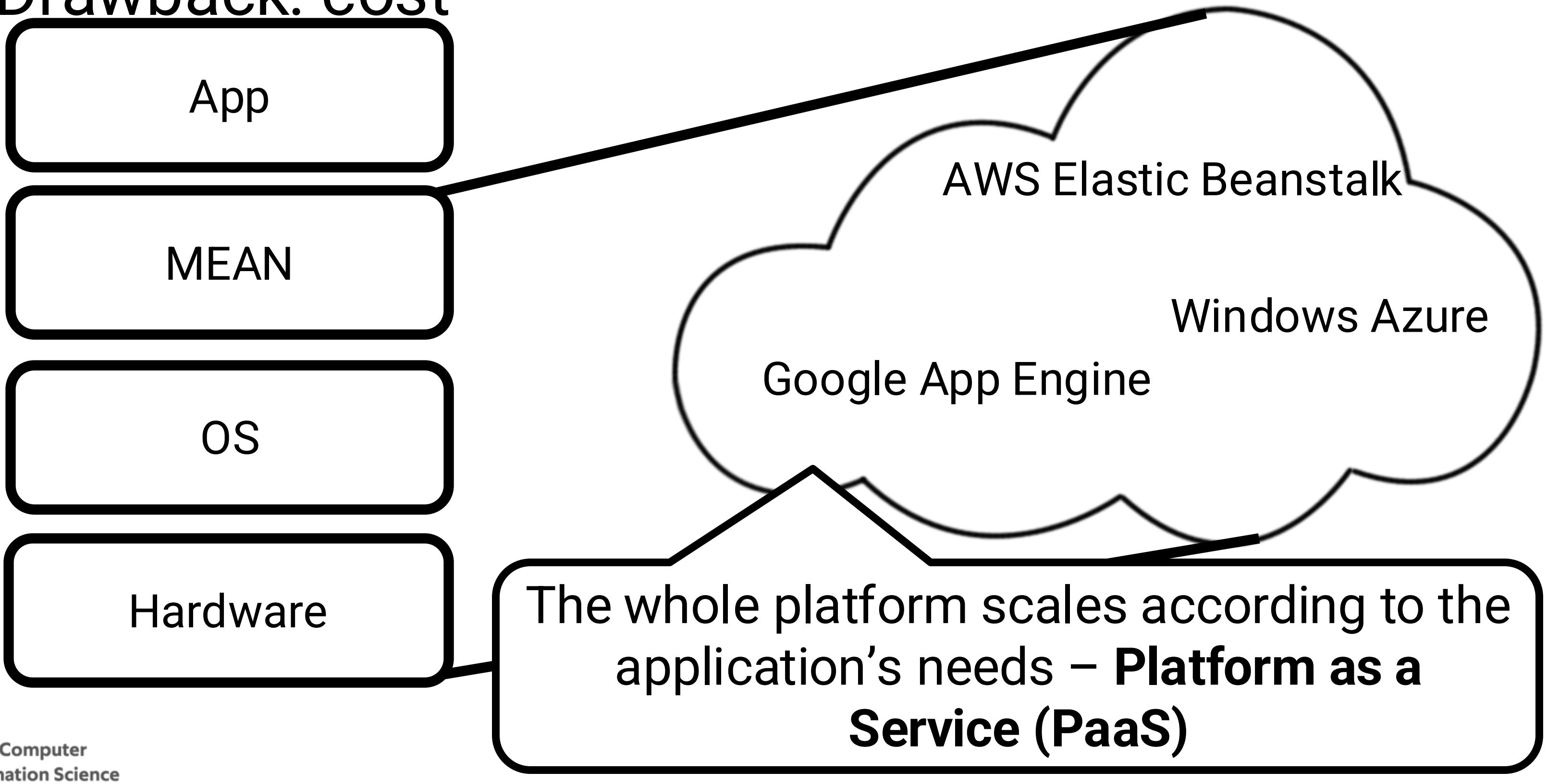
- MEAN is like a middleware
  - Web applications on top
  - Different options for operating systems below



Cloud computing enables “hardware” to scale according to the application’s needs – **Infrastructure as a Service (IaaS)**

Web Platform – Current Standard

- Idea: why not virtualise the whole platform!
  - Benefit: the developer can concentrate on the app
  - Drawback: cost



App

MEAN

OS

Hardware

AWS Elastic Beanstalk

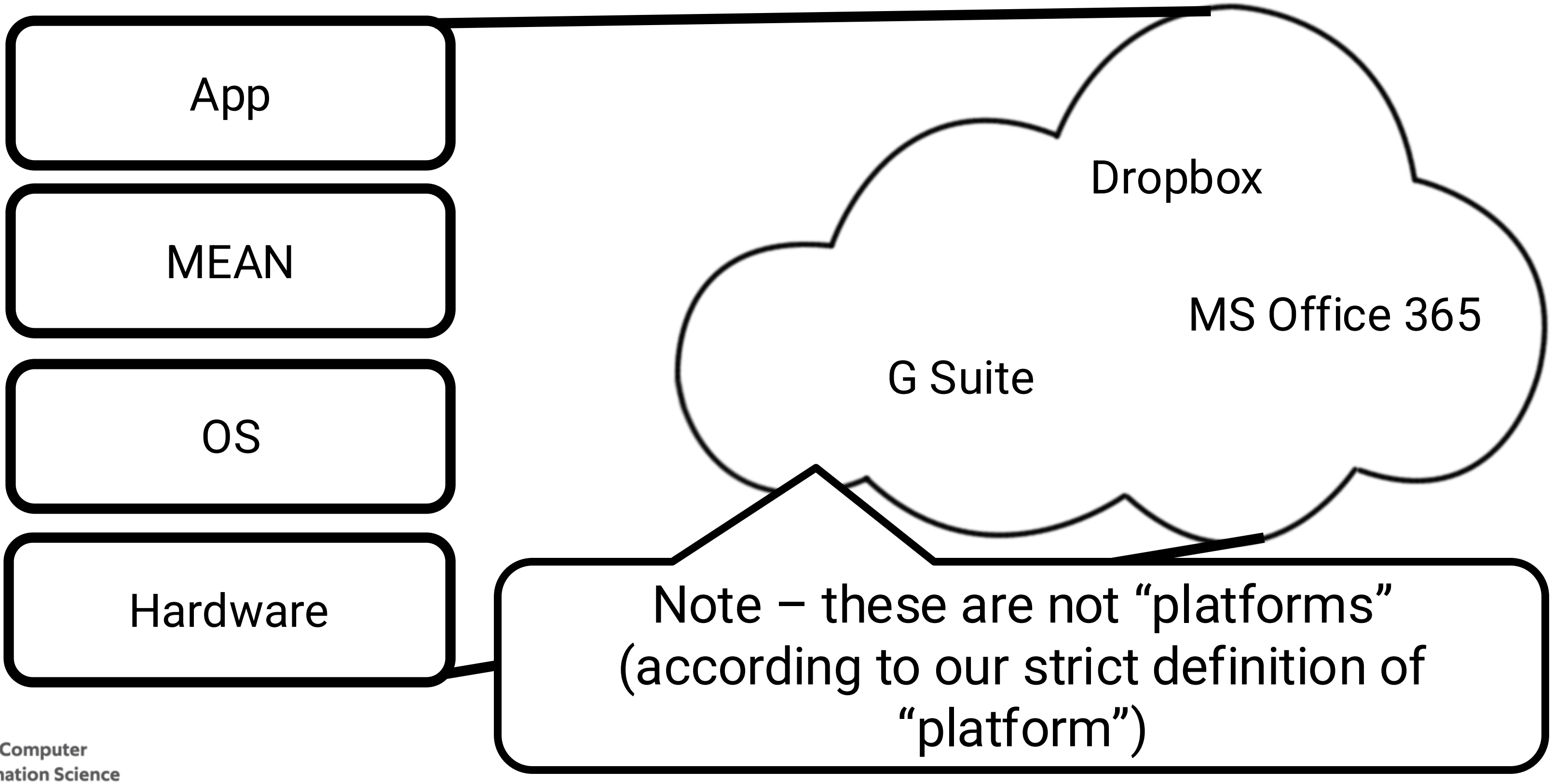
Windows Azure

Google App Engine

The whole platform scales according to the application’s needs – **Platform as a Service (PaaS)**

Web Platform – Current Standard

- Step further – Software as a Service (SaaS)



Note – these are not "platforms" (according to our strict definition of "platform")

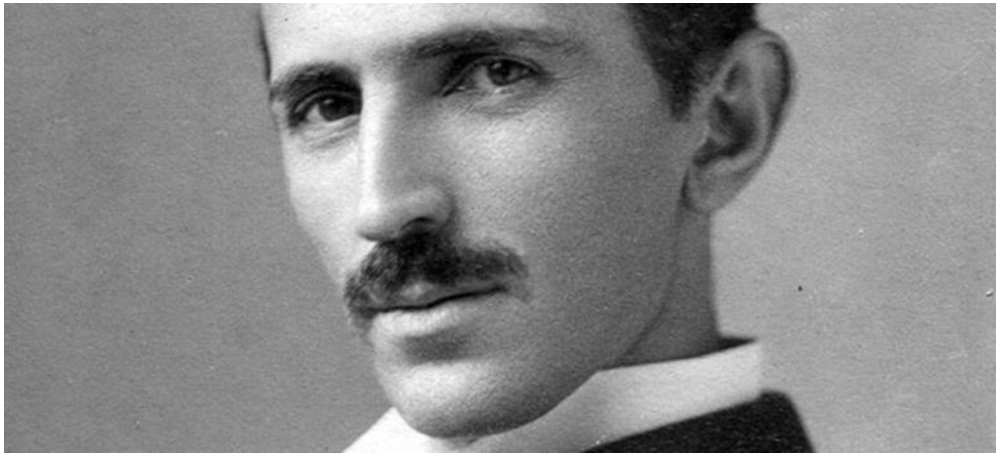
Web Platform – Conclusions

- Evolution towards the 1945 vision of associatively connected information
- HTTP and the Internet enable WWW
- Server-client architecture initially decoupled backend and frontend programming
- MEAN architecture enables both with a single programming language – JavaScript
- Cloud computing enables scalability
- TODO: think about constraints of the current Web platform

Could it have evolved in a different direction?

Mobile Computing Platforms

Vision of Mobile Computing



"When wireless is perfectly applied the whole earth will be converted into a huge **brain**, which in fact it is, all things being particles of a real and rhythmic whole. We shall be able to communicate with one another instantly, irrespective of distance. Not only this, but through television and telephony we shall see and hear one another as perfectly as though we were face to face, despite intervening distances of thousands of miles; and the instruments through which we shall be able to do his will be amazingly simple compared with our present telephone. A man will be able to carry one in his vest pocket."

Nikola Tesla, 1926

Vision of Mobile Computing

"Communications will become sight-sound and you will see as well as hear the person you telephone. The screen can be used not only to see the people you call but also for studying documents and photographs and reading passages from books." And more: "The appliances of 2014 will have no electric cords, of course, for they will be powered by long-lived batteries running on radioisotopes." Isaac Asimov, 1964

Vision of Mobile Computing

“The wristwatch telephone will be technologically feasible very soon. [..] It will be completely mobile. And this would again restructure society.

You'll tell the machine I'm interested in such-and-such item, sports, politics and so forth, and the machine will hunt the main central library and bring all this to you selectively. Just what you want, not all the junk [...] The newspaper is on the way out. We're not gonna ship all this tons and tons of paper around when all you need is information.

And of course it has disadvantages as well as advantages. Anyone can get at you anytime you like. Of course you could switch off the calling signal, but then you might have to explain later why it was switched off.”

Arthur Clarke, 1976

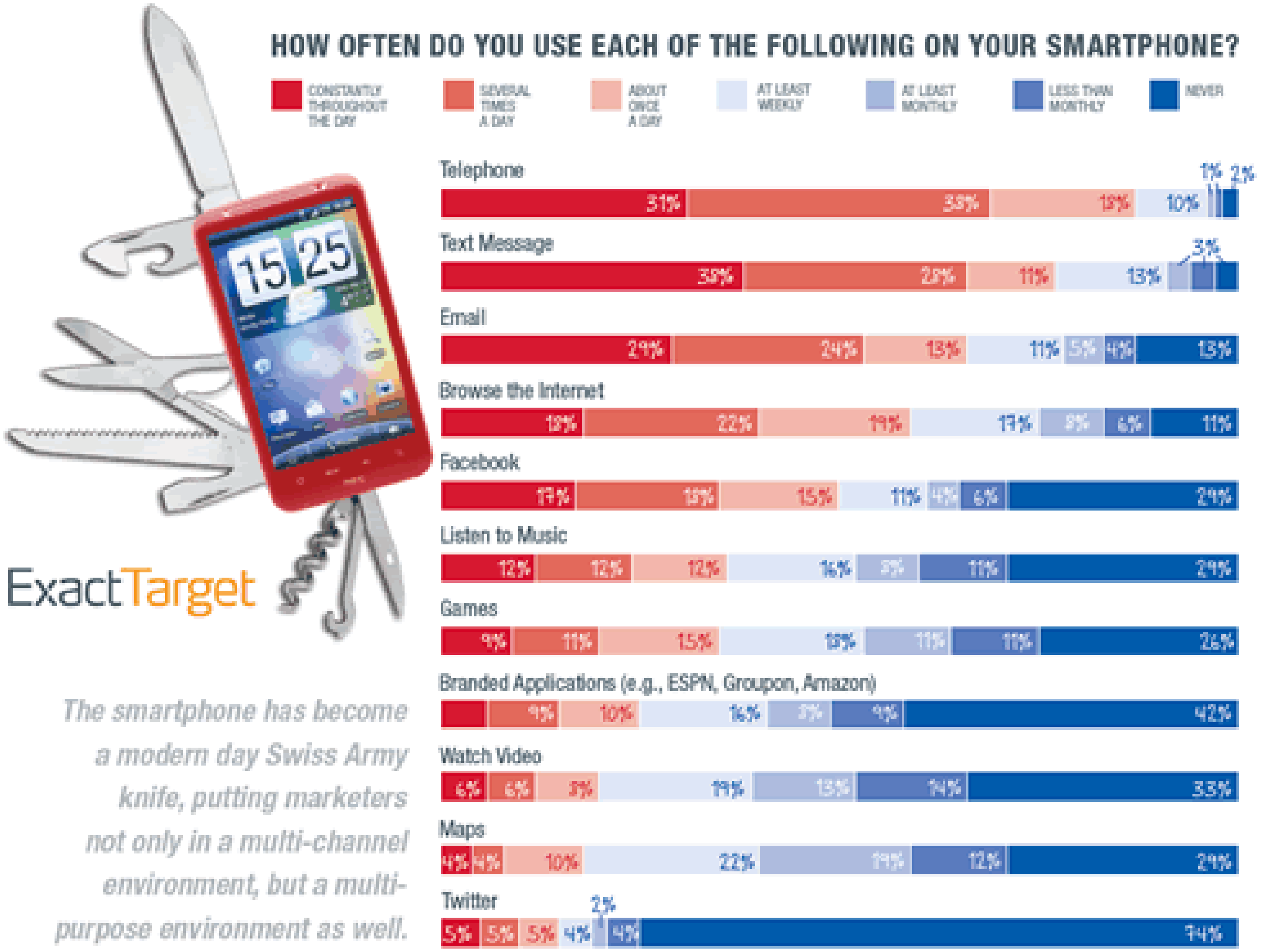
Mobile Computer Development Timeline

- 1994 – IBM Simon
  - Cellular telephony + emailing, faxing + apps
- Late 1990’s – PDAs
  - Nokia 9000 Communicator
  - Early versions of mobile operating systems
- Early 2000s – Mass adoption
  - Blackberry, DoCoMo in Japan

Mobile Computer Development Timeline

- 2007 – iPhone
  - Touchscreen, sensors, App Sto
- 2008 – Android OS
  - Open source, Linux-based
  - Runs on a large number of devi
- Today
  - More than 3 billion users
  - Dynamic market (Samsung, Xiaomi, Huawei, etc.)
  - Millions of apps

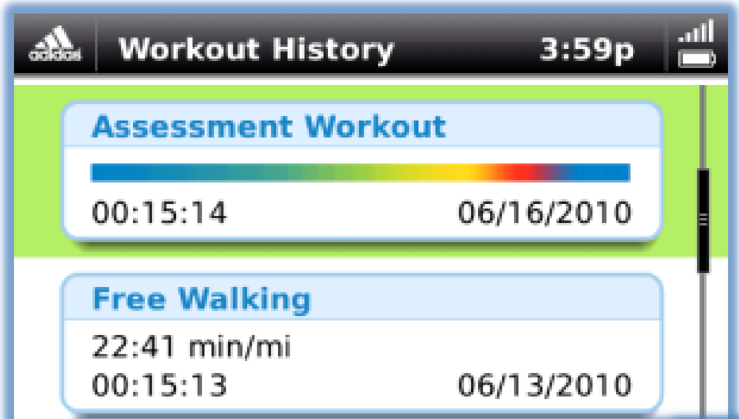
Digital Swiss Army Knife



The smartphone has become a modern day Swiss Army knife, putting marketers not only in a multi-channel environment, but a multi-purpose environment as well.

Smartphone Use Cases

Personal trainer



Workout History 3:59p

Assessment Workout

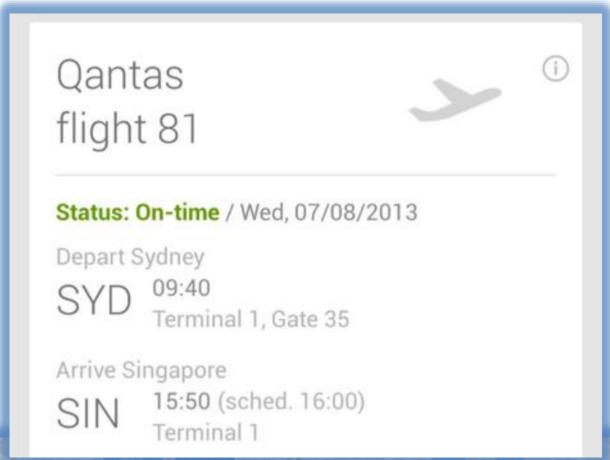
00:15:14 06/16/2010

Free Walking

22:41 min/mi

00:15:13 06/13/2010

Proactive assistant



Qantas flight 81

Status: On-time / Wed, 07/08/2013

Depart Sydney

SYD 09:40 Terminal 1, Gate 35

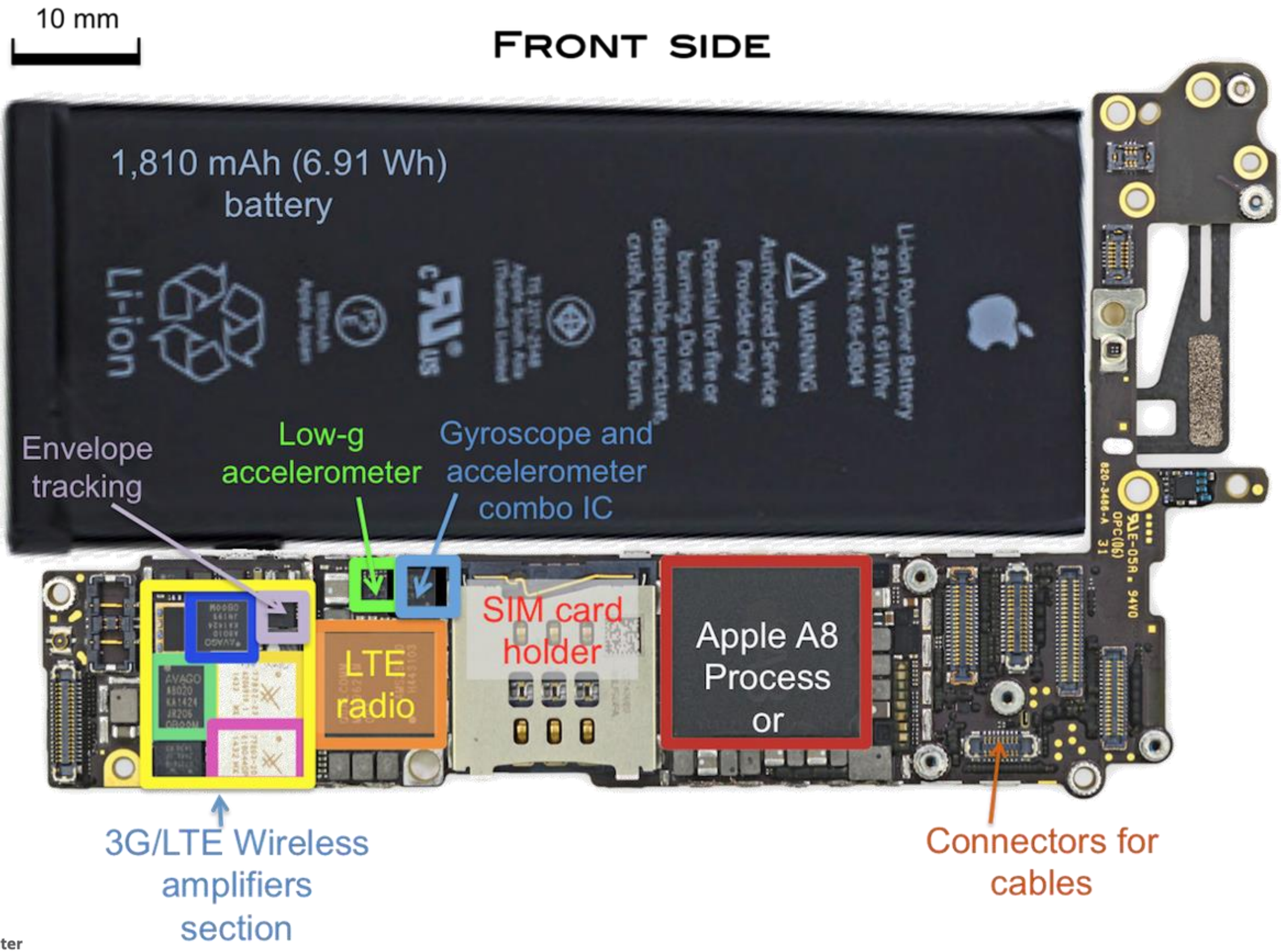
Arrive Singapore

SIN 15:50 (sched. 16:00) Terminal 1

Medical device

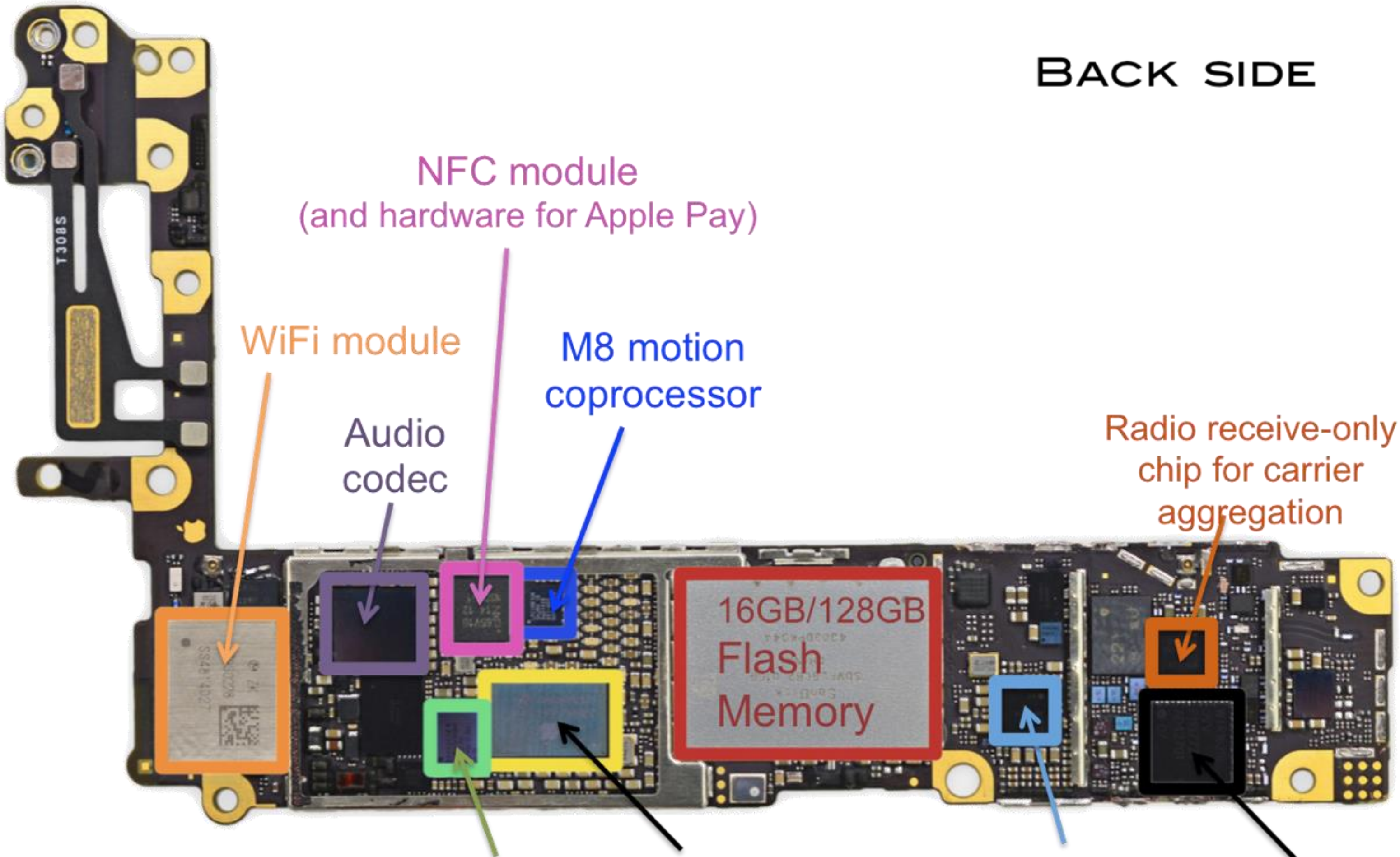
Location-aware online social network

Mobile Platform – Hardware



Mobile Platform – Hardware

Back side



NFC module
(and hardware for Apple Pay)

WiFi module

M8 motion coprocessor

Audio codec

Radio receive-only chip for carrier aggregation

16GB/128GB Flash Memory

Touchscreen controller

Primary Power Management IC

Secondary Power Management IC

RF transceiver

Mobile Platform – Operating Systems

- Symbian – started in 1998 (now discontinued) used by Nokia, Samsung, Sony Ericsson, etc.
  - Resource efficient OS that worked on multiple hardware platforms
  - Why did it fail then?
- Smartphones brought new affordances, the OS has to support personalised use and developers' (crazy) ideas
  - UI is very important
  - The utility of the device is measured by the size of its developers' community

Mobile Platform – Operating Systems

- iOS – created by Apple for iPhone, iPad, iPod
  - Based on the same core as the desktop OS (Mac OS X) – Darwin
  - Runs on iPhones – specific hardware, thus iOS is not transplantable to other hardware platforms
- iOS development supported through iOS Software Development Kit (iOS SDK)
  - Access to sensors, networking, processing, etc.
  - Cocoa Touch UI framework – so that all iOS apps have the same look and feel
- Programming in Objective-C or Swift

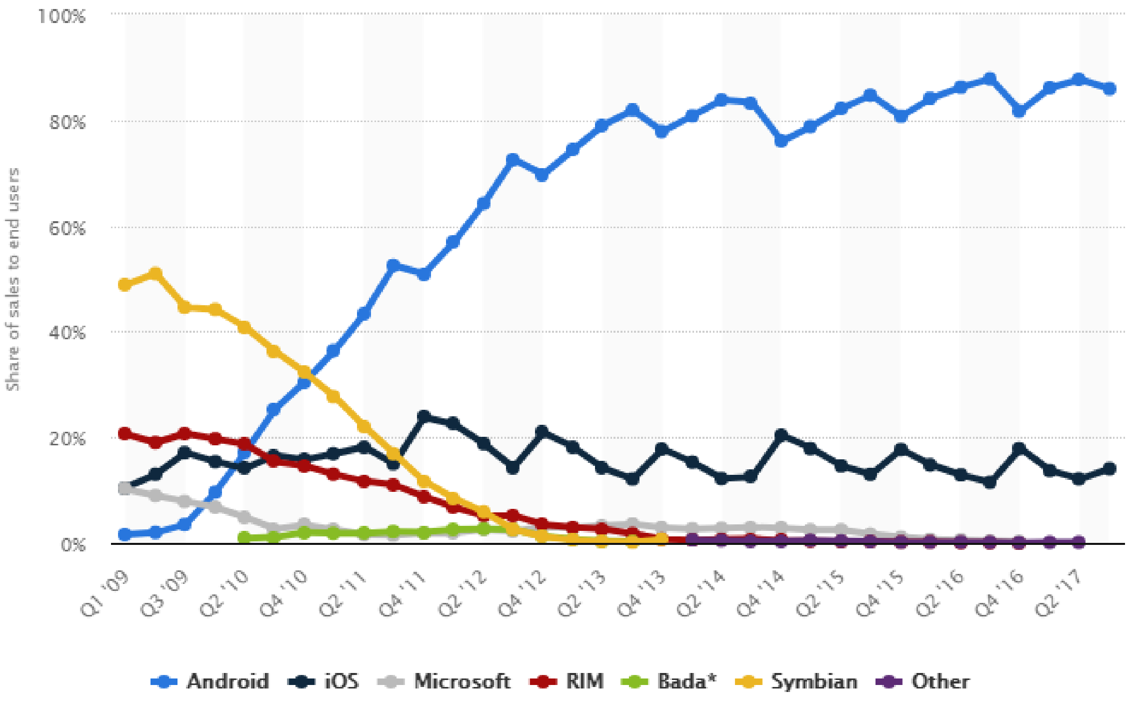
Mobile Platform – Operating Systems

- Android –created by Google for a wide range of mobile devices
  - Based on Linux kernel
  - Runs on different hardware platforms
    - In practice, some API calls give unexpected results on certain devices
- Development supported through Android SDK and Android Studio
  - Access to sensors, networking, processing, etc.
  - Guidelines for UI implementation – Material Design
- Programming in Java, Kotlin, C++

Mobile Platform – OS Market Share

- Android dominates the market
  - Google's brilliant, yet ethically questionable business strategy

Give us your data, everything else is free!



© Statista 2018

Mobile Platform – Conclusions

- Mobile phones are personal devices and user interaction remains crucial – the platform should focus on seamless interaction
- Mobile phones are used for various purposes – the platform should allow easy application development and distribution
- Context awareness makes mobiles adaptive to the current situation – the platform should enable access to sensors (e.g. accelerometer) or even higher-level inferences (e.g. a user's physical activity)

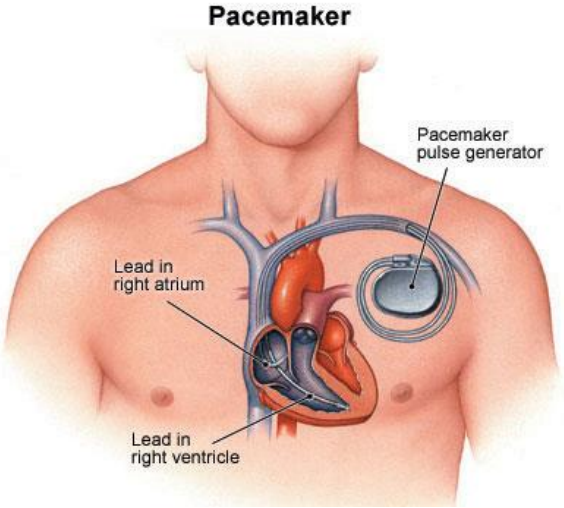
Embedded Platforms

Embedded Systems

- Computer systems that can be embedded in (everyday) objects
- Around 98% of computers are embedded
- “31 billion microcontrollers shipped in 2021”

https://www.statista.com/statistics/935382/worldwide-microcontroller-unit-shipments

Embedded Systems



Embedded Systems

- Computer systems that can be embedded in (everyday) objects
- Around 98% of computers are embedded
- 31 billion microcontrollers sold in 2021
- Let’s try to approximate!
  - ~100 billion?
- Compare to:
  - 2 billion PCs and 5 billion smartphones

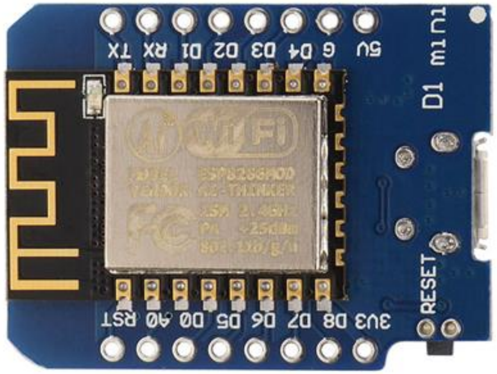
Embedded System Programming

- Very constrained systems (energy, autonomous operation, real-time operation, etc.)
- Interact with the environment through sensors
- React to the environment through actuators
- Need to communicate via wireless

More about (distributed) embedded systems next week (read Estrin et al. piece!)

Embedded System Programming

- Memory and energy constraints call for low-level programming
  - C/C++, assembler
  - Micro Python, uJ

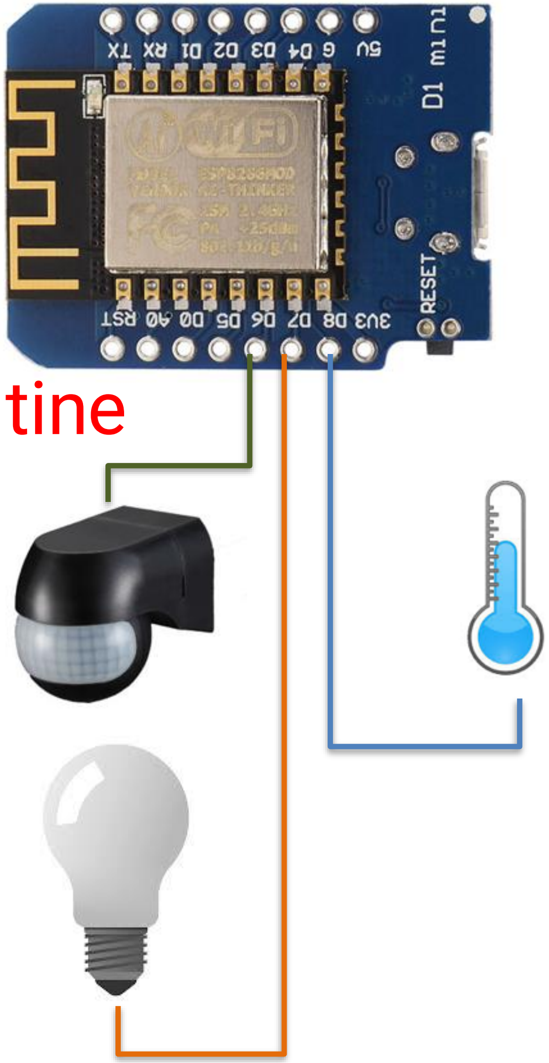


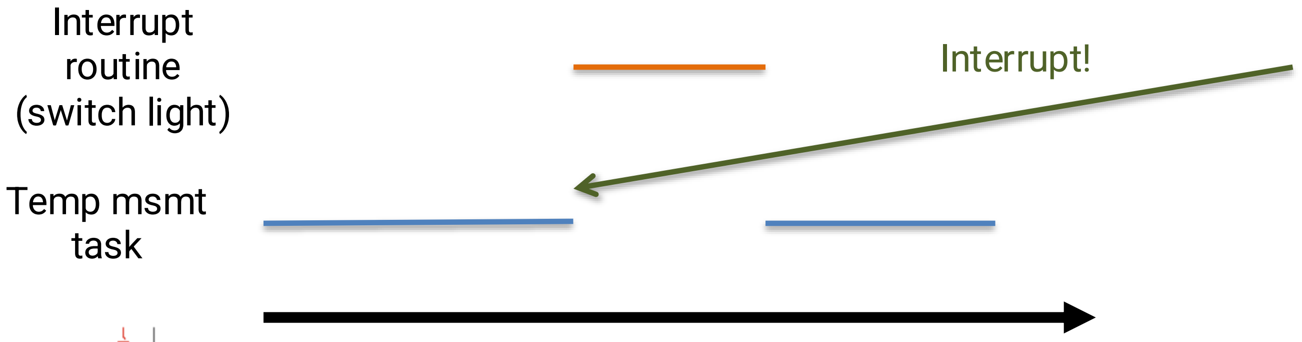
- Task completion time constraints call for real-time operating systems with guaranteed task handling times
  - Event-driven pre-emptive multitasking
    - Interrupts
  - Time-sharing

Car anti-blocking break system needs to react “now” even if the microcontroller is processing something else at the moment

Embedded System Programming

- A low priority task (temp msmt) is executed in an infinite loop
- A high priority event (movement detected) preempts the current task and executes the interrupt routine





Embedded Platform

We will use Arduino in the labs!

- Operating system support:
  - None (Arduino – only a bootloader that loads the firmware you write to the microcontroller)
  - Real-time OS (FreeRTOS – a microkernel)
  - Specific purposes (TinyOS – a monolitic kernel optimised for low-power wireless sensor networks)
- Communication with hardware components through protocols (I2C, SPI)
- Communication with other devices through wireless protocols (Bluetooth, NFC, RFID)

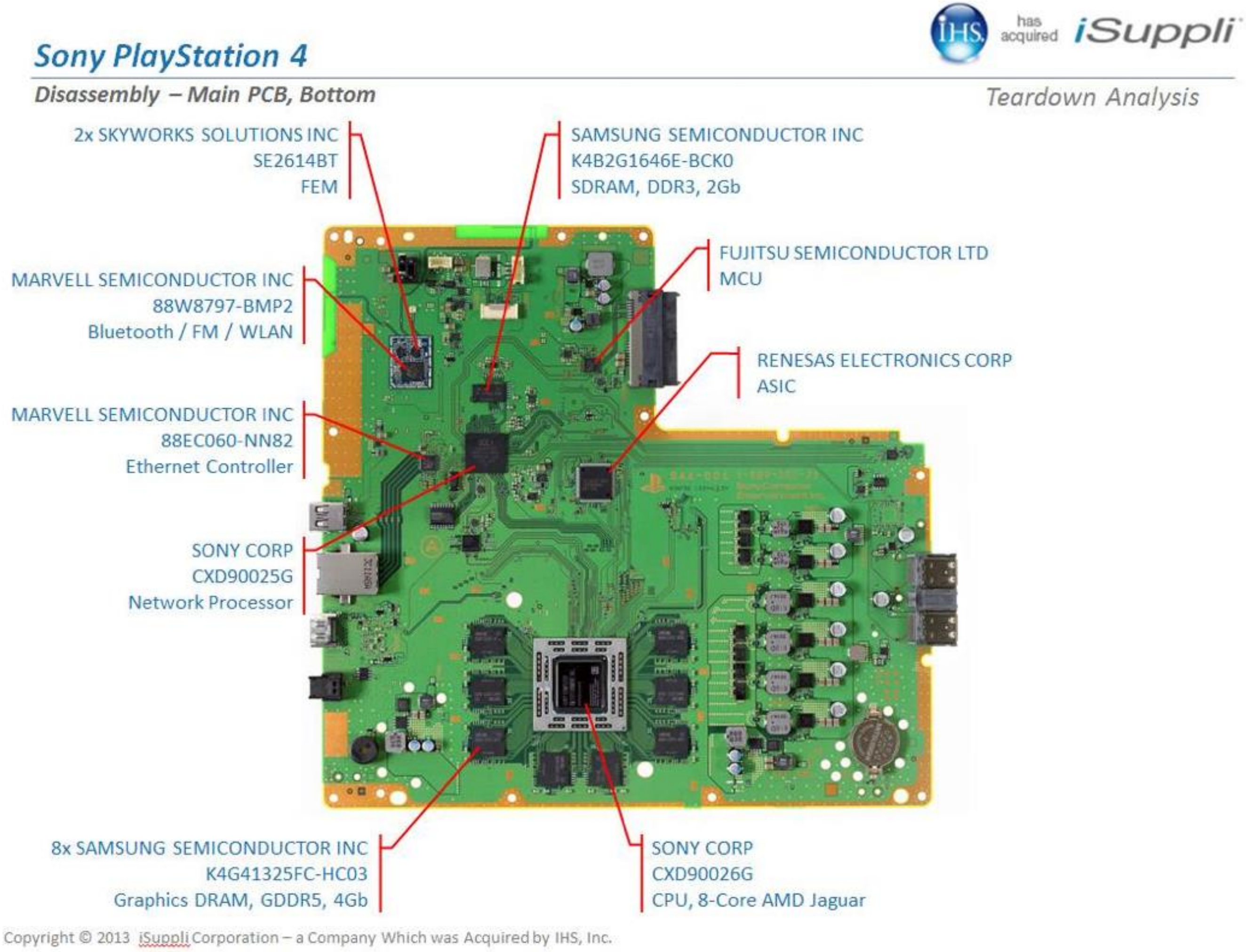
Gaming Platforms

Gaming Platforms

- From embedded systems to high-processing power computers

- Need for high-quality graphics
  - Hardware: Graphical processing units (GPUs)
  - Software: Game engines

Gaming Hardware



IHS has acquired iSuppli

Sony PlayStation 4

Disassembly – Main PCB, Bottom

Teardown Analysis

2x SKYWORKS SOLUTIONS INC
SE2614BT
FEM

SAMSUNG SEMICONDUCTOR INC
K4B2G1646E-BCK0
SDRAM, DDR3, 2Gb

MARVELL SEMICONDUCTOR INC
88W8797-BMP2
Bluetooth / FM / WLAN

FUJITSU SEMICONDUCTOR LTD
MCU

RENESAS ELECTRONICS CORP
ASIC

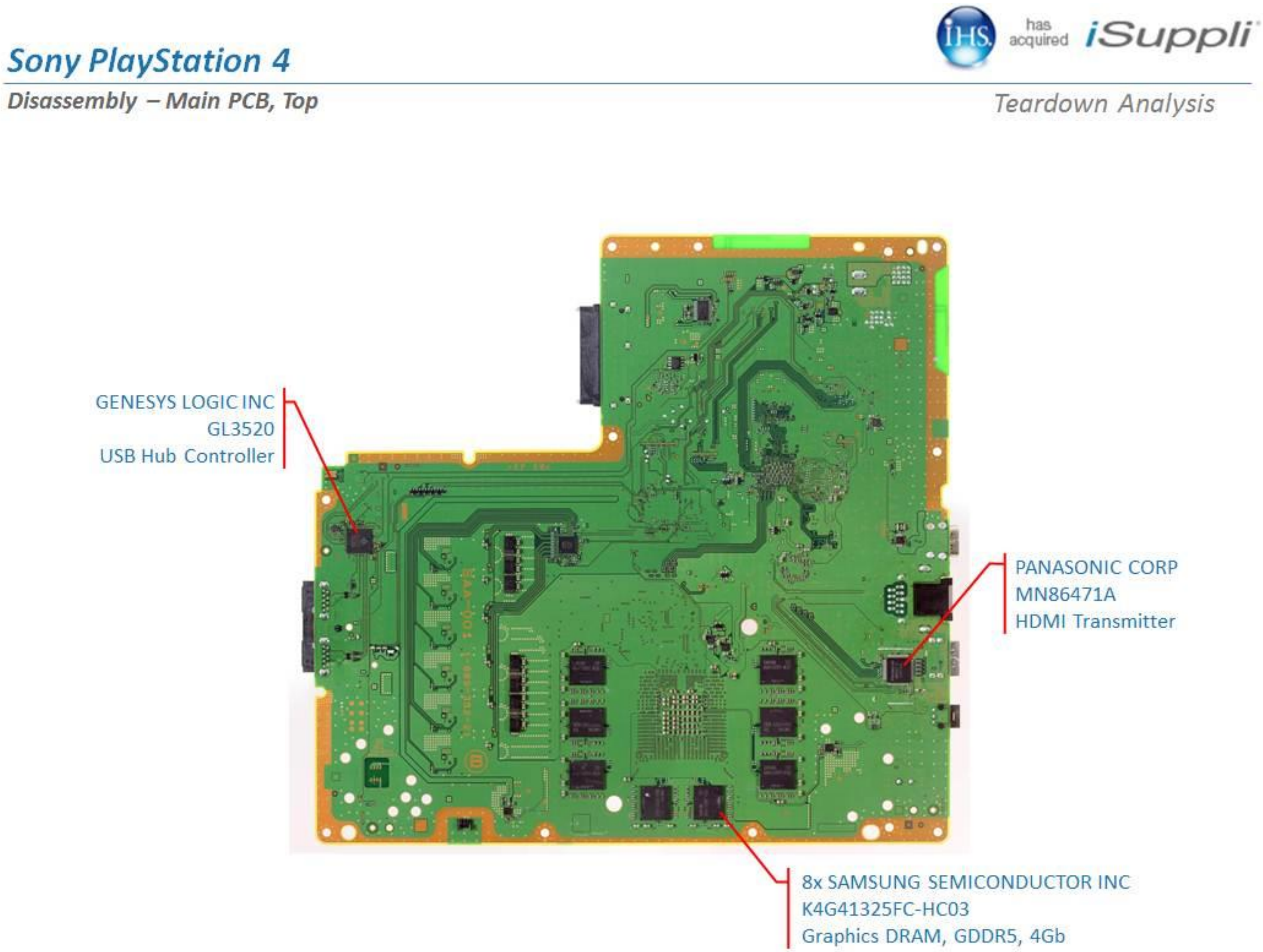
MARVELL SEMICONDUCTOR INC
88EC060-NN82
Ethernet Controller

SONY CORP
CXD90025G
Network Processor

8x SAMSUNG SEMICONDUCTOR INC
K4G41325FC-HC03
Graphics DRAM, GDDR5, 4Gb

SONY CORP
CXD90026G
CPU, 8-Core AMD Jaguar

Gaming Hardware



Sony PlayStation 4

Disassembly – Main PCB, Top

Teardown Analysis

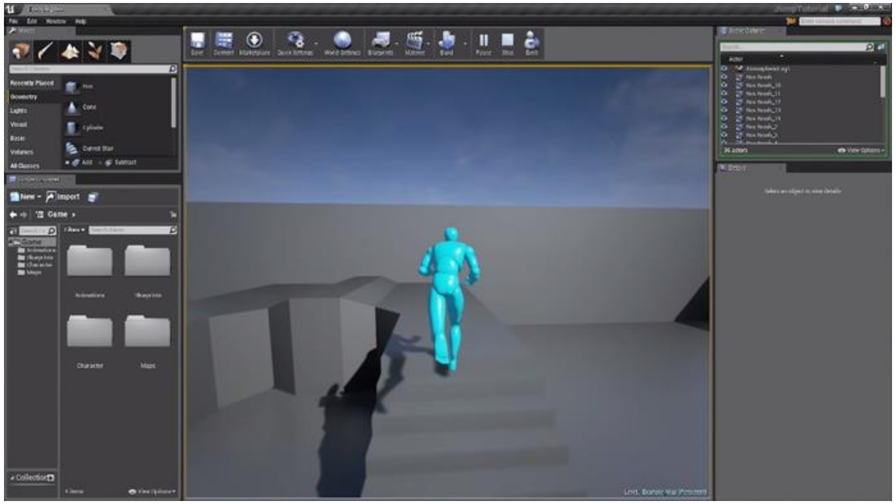
GENESYS LOGIC INC
GL3520
USB Hub Controller

PANASONIC CORP
MN86471A
HDMI Transmitter

8x SAMSUNG SEMICONDUCTOR INC
K4G41325FC-HC03
Graphics DRAM, GDDR5, 4Gb

Gaming Software Support

- Gaming engines support game development:
  - Rendering
  - Physics
  - Audio
  - Artificial intelligence



- Development in high-level languages e.g. C#/JavaScript for Unity UnrealScript for Unreal engine

Platform Overview Conclusions

- The evolution of computing led to differentiation of user expectations
- General purpose computers cannot satisfy these diverse expectations
- Different platforms answer to different needs but come with different constraints and programming paradigms
- TODO:
  - Read the syllabus carefully
  - Read Tim Berners-Lee's and Estrin et al. article