G53MDP
Mobile Device Programming

Mobile Phone Architecture

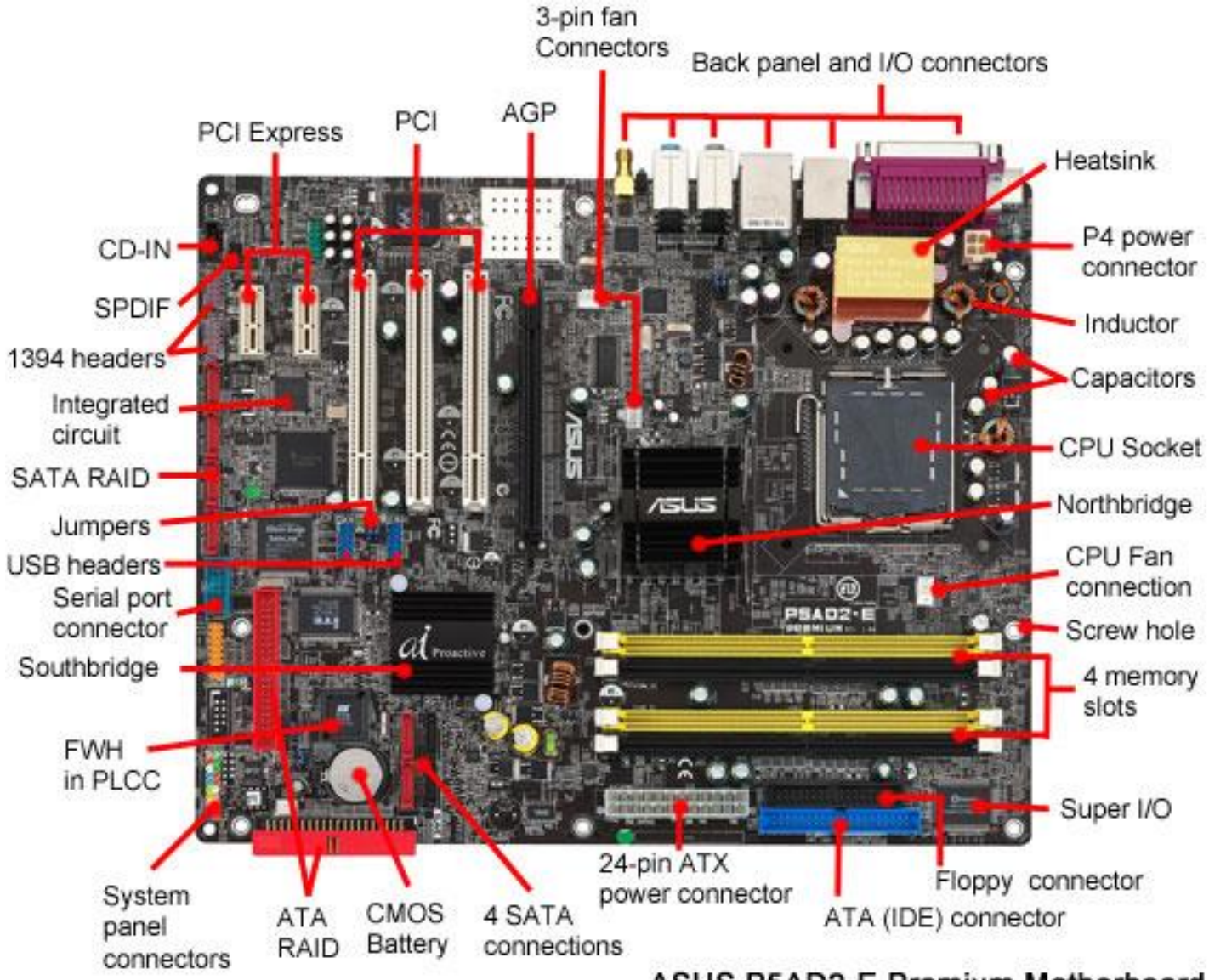
Iphone X vs Galaxy S8 vs Inspiron 13

	Iphone X	Galaxy S8+	Inspiron 13
CPU	Apple A11 Bionic 2xMonsson+ 4xMistral	Qualcomm Snapdragon 835 4x2.3 GHz + 4x1.7GHz	**8th Gen Intel i5-8250U**
OS	IOS 11	Android 7.0 (Nougat)	**Windows 10**
GPU	Apple 3-core	Mali-G71 MP20	**Intel UHD Graphics 620**
RAM	3G	6G	**8G**
Internal Memory	256GB	128 GB	**256 GB (SSD)**
Size /Weight	78 cm^3/174g	95 cm^3/173g	1457 cm^3/1620g
Sensors	Accelerometer; Gyroscope; Compass; Barometer; FaceID/FingerID		--
Price	£ 1149	£ 779	**£ 749**

Learning Outcomes

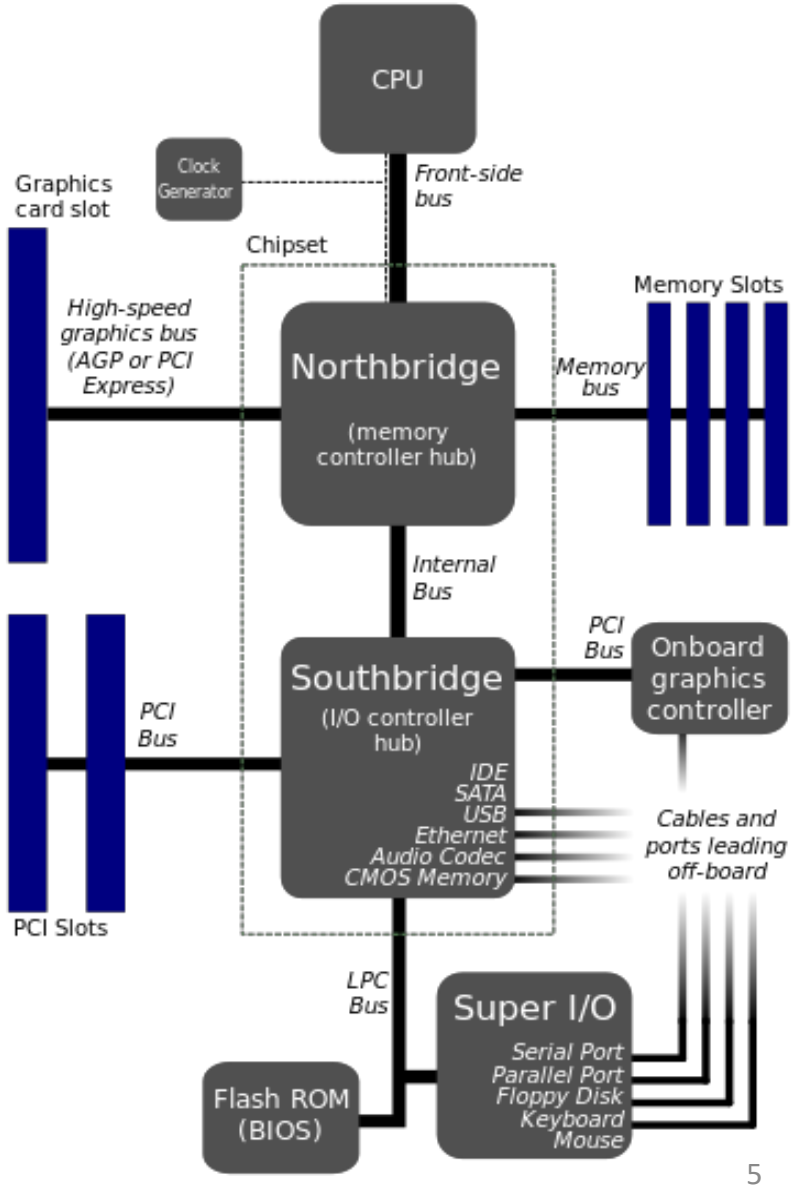
- Understand the architectural differences between a PC and a mobile phone
- Have an overview of the main components of a mobile phone
- Understand some of the differences between ARM and x86 as relevant to mobile phones

A Typical Motherboard of a PC



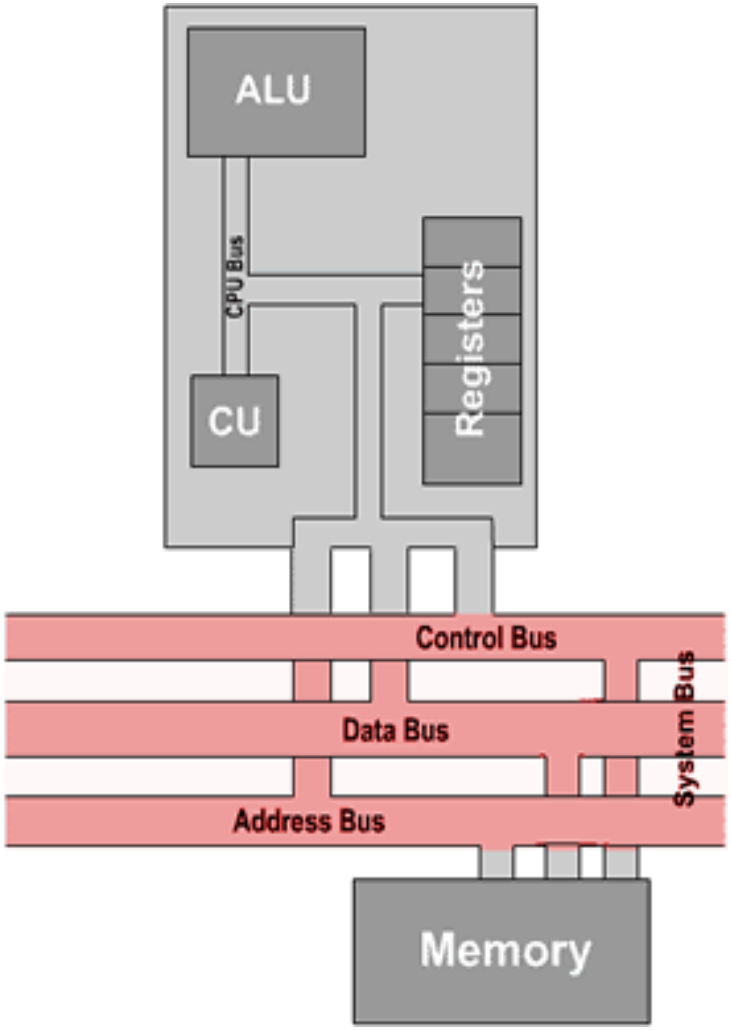
Main Components

- Multiple chips
- CPU
- Northbridge
- Southbridge
- RAM
- Disks (i.e. HDD/SSD)
- Several functional cards (NIC, GPU, Audio, etc.)



Connections

- CPU has a bus that connects it to the other devices / chips (address bus, data bus & control buses)
- RAM chips etc. have similar connections
- Connect CPU to RAM / ROM
- Add some control logic and you have a PC

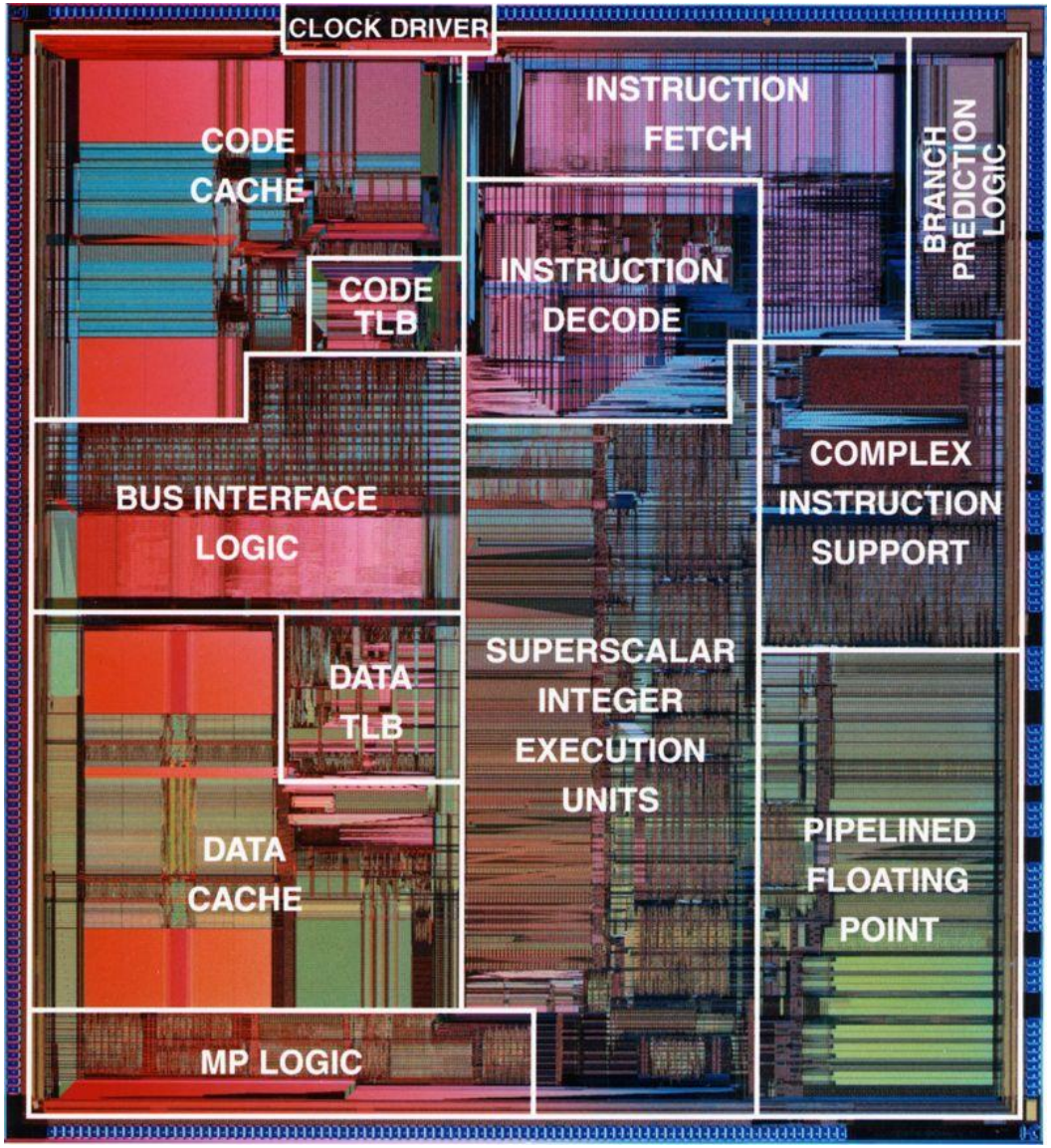


Connections

- Nowadays more advanced buses are used
  - E.g. HyperBus
- Bus width
  - More wires allows more data to be transferred ~= faster bus
  - More pins results in larger chip size
- Multiple chips and connections
  - Communications overhead / voltage drop = power
  - Clean architectural principle

Fancy a pocket PC ?

Intel Pentium Chip (1993)



Making better use of transistors

- A CPU is a collection of transistors
  - All digital logic devices are built out of transistors
  - 4 transistors build a NAND gate (Any logic circuit can be built using multiple NAND gates)
  - Moores law: number of transistors per square inch doubled every two years.
- Only use some of the transistors on a chip to form the CPU
  - Use the rest to build the other components of the system
  - External pins connect directly to peripheral hardware
- Called System on a Chip (SoC)
  - The whole system is literally on a single chip
    - Integrates several heterogeneous components on a chip
  - Aims to reduce communications overhead

System on a Chip

- Transistors provide computation, storage
  - Divide the chip into multiple communicating regions
- Build block by block from descriptions of separate parts
  - IP (intellectual property) cores
    - CPU core
    - GPU core
    - RAM core

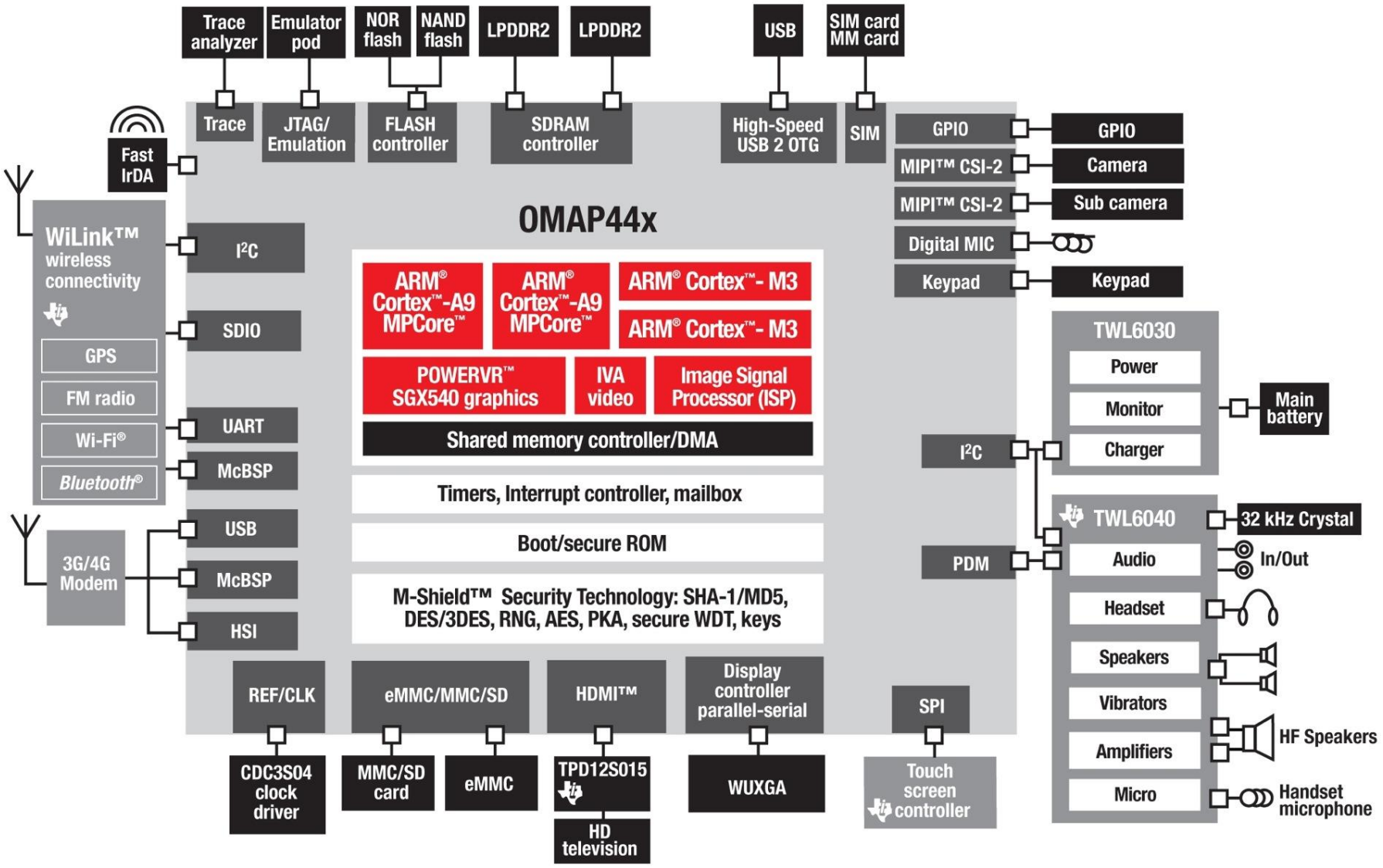
System on a Chip

- Qualcomm Snapdragon
  - Google, Motorola, Samsung
- Apple A4-A11
  - iPhones
- Texas Instruments OMAP
  - Kindle Fire, Nook
- All share the same CPU block
  - ARM Cortex A*
- But may have different companion blocks

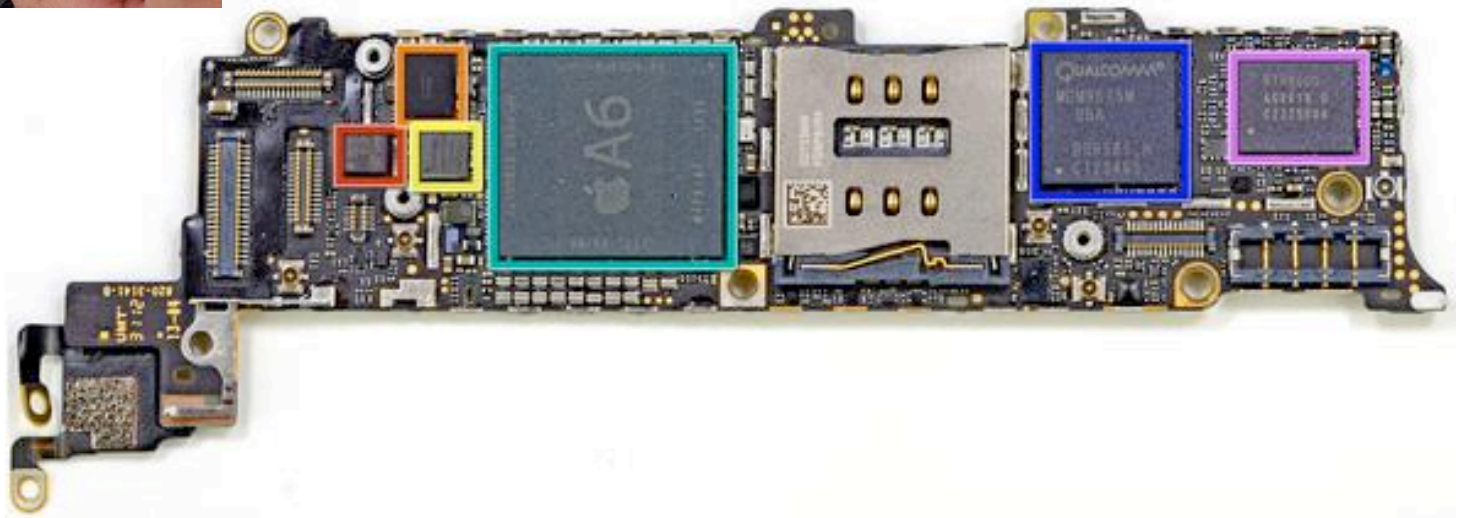
OMAP 44x0

- A typical SoC chip used in mobile phones
- Motorola Droid Bionic, Samsung Galaxy, Nexus
- Two ARM Cortex-A9 CPU
- 3D GPU (PowerVR SGX)
- Image Video Audio (IVA) Accelerator
- Image Signal Processor

OMAP 44x0 Layout



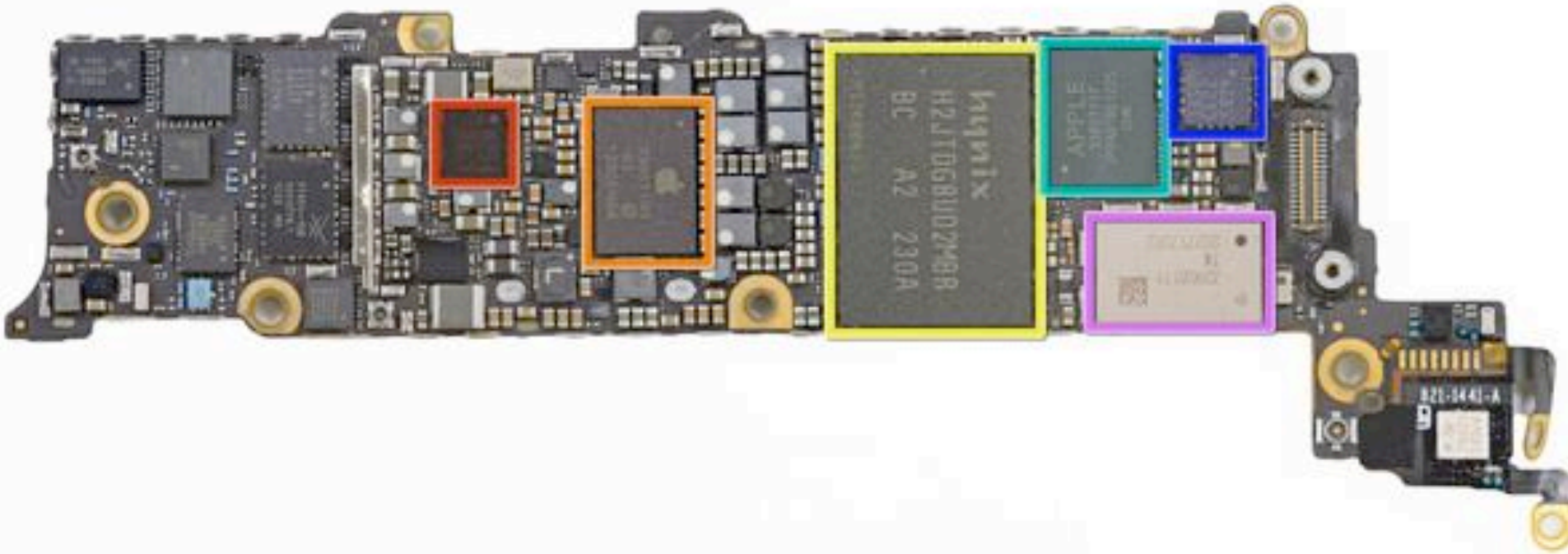
Inside an Iphone5/5C



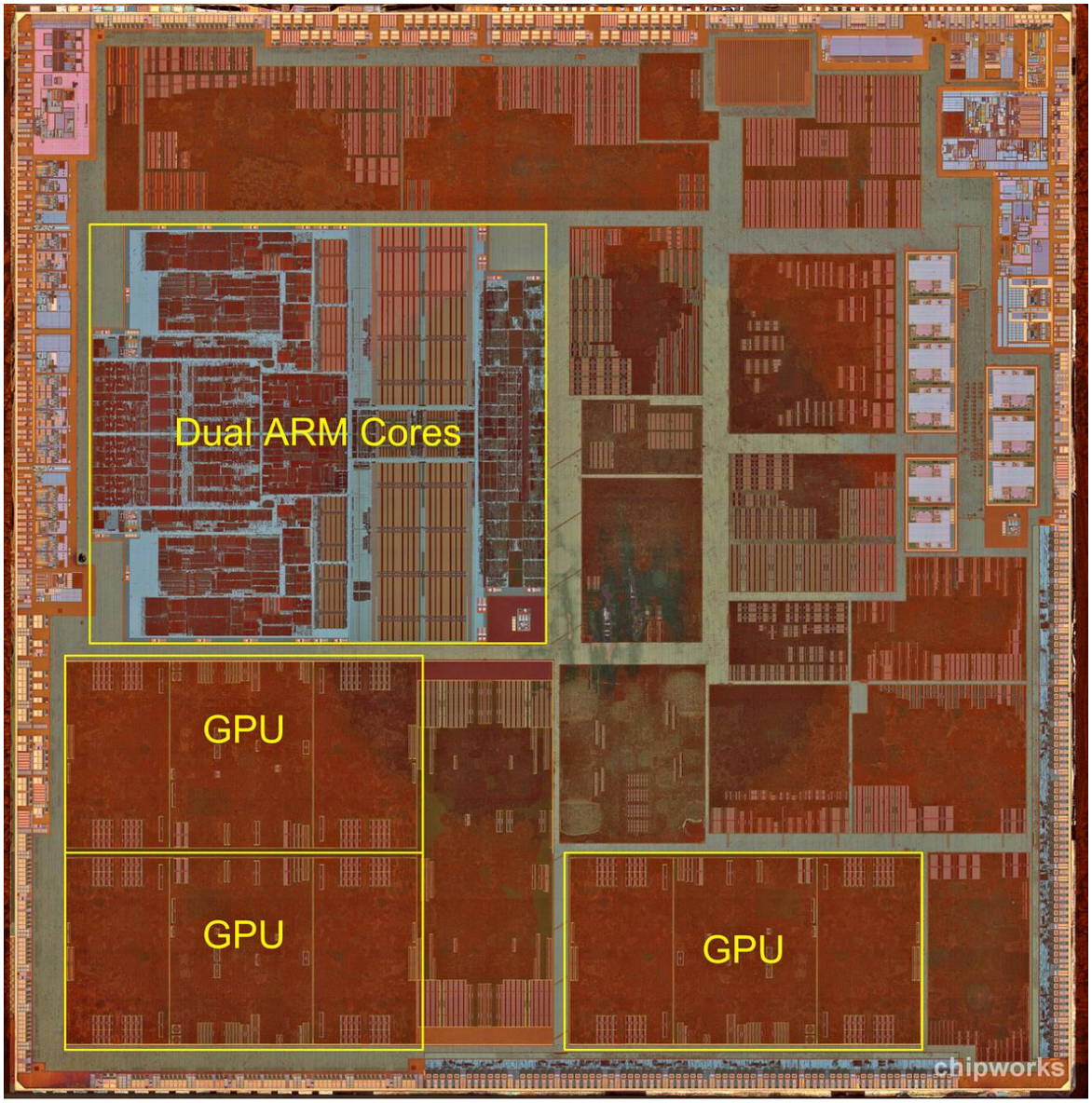
Some analogue components have a limited minimum size (e.g. RF transceiver, Modem, etc.)

Inside an Iphone5/5C

Wifi, power, storage, audio…



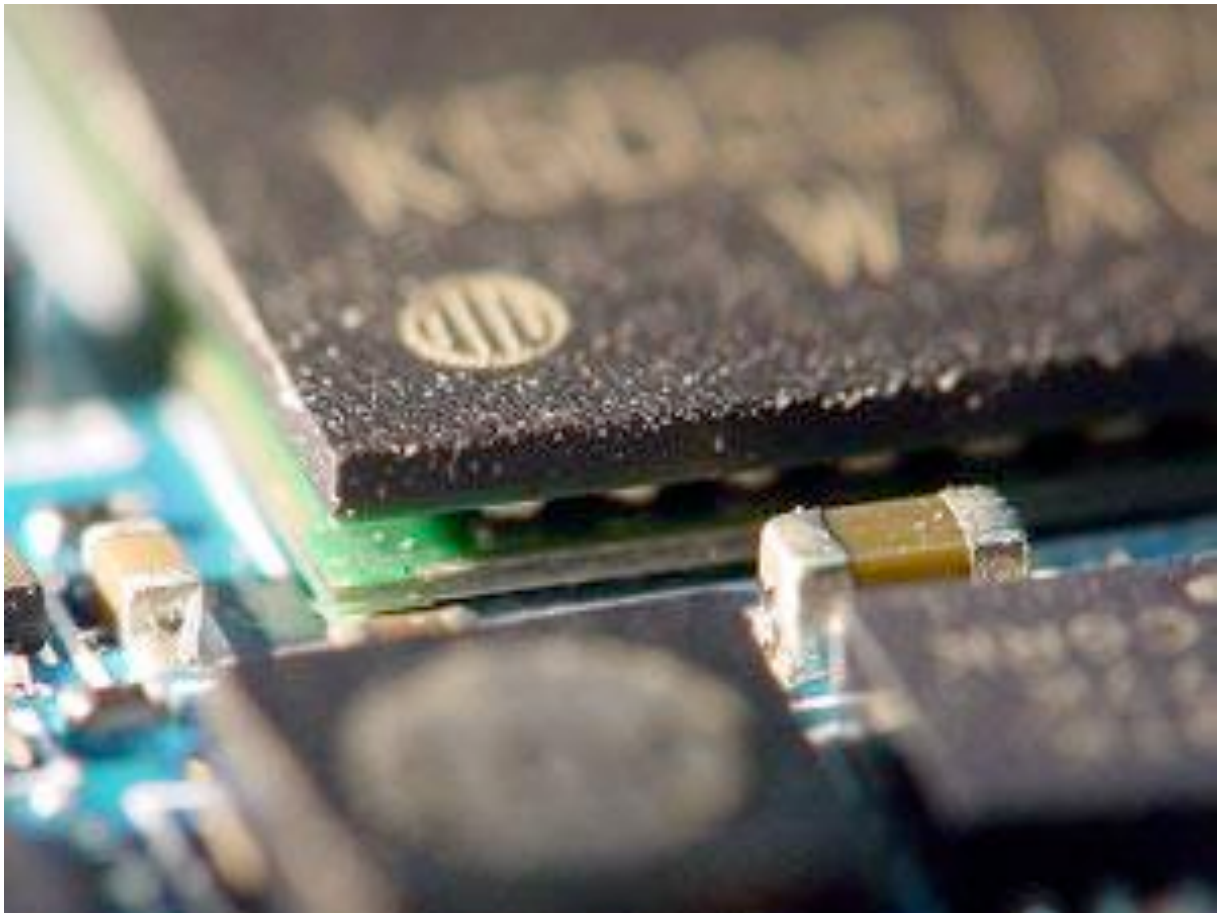
Internal Structure of A6

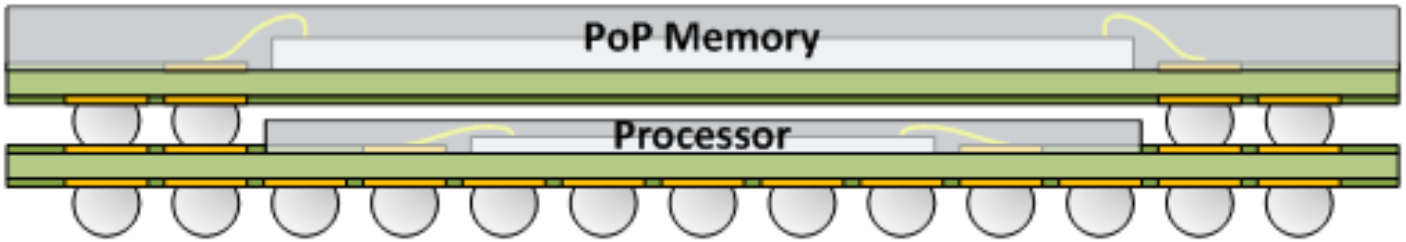


Package-on-Package

- RAM is normally not included in SoC
  - Uses a lot of space
- Separate package on top of the SoC
- Called Package-on-Package (PoP)
  - Separate memory and logic production
  - “Stack” packages to minimise space
- RAM are shared between multiple components (e.g. CPU and GPU)
- No swap partition / page file
  - Hard limit
  - Code should not consume a lot of RAM.

Package-on-Package





ARM CPU

- Almost all (95%) of smart phones use ARM CPUs
- Sold as a design, not a physical device
  - Great for SoC usage
  - Chip manufactures can add ARM CPU to bespoke SoC vs buying a chip from Intel
- 32-bit / 64-bit CPU
  - Multiple instruction sets
  - Older phones made use of Jazelle DBX / embedded systems
    - Run Java bytecode directly, dedicated chips
- Aims
  - Fast and efficient
    - Good for mobile applications
    - Best use of battery – more instructions = more battery use
  - High code density
    - Less moving stuff around, best use of space

ARM CPU

- RISC-design (Reduced Instruction Set Chip)
- Only includes simple instructions that can be executed in one clock cycle
  - More efficient decoding
- Remove instructions you might expect in x86
  - e.g. divide instructions
  - If need to divide, implement in software
  - Or rather, the compiler does it for you

ARM CPU

Register names

r0	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11	r12	r13	r14	r15

- 16 registers
- Load/Store architecture
- Each instruction is 32-bits long
- Makes decoding easy / efficient
  - Vs x86 variable instruction length
- But means loading a constant into a register can be tricky
  - Constants must fit into the 32-bit instruction width
    - Can't therefore be the full 32-bits
  - 8-bit + a 4-bit shift — allows more complex operations
  - Also a Move Negated instruction
  - Otherwise, use literal pool

Opcode	Type	Decoded Instruction
`0xE3A00000`	Data Processing	`MOV R0, #0`
`0xEF000002`	Software Interrupt	`SWI 2`
`0xEAFFFFFC`	Branch Instruction	`B -16`
`0xE0810002`	Data Processing	`ADD R0, R1,R2`

ARM Speed

- Does running at 1GHz means its going to be slow?
- GHz-speed tells us ‘cycles per second’
  - Actual speed depends on how many cycles an instruction takes
  - ARM aims for one-cycle per instruction
- x86 instructions can take many cycles
  - If an ARM instruction takes 1 cycle at 1GHz
  - And an x86 instruction takes 3 cycles at 3GHz
  - Which is faster?
- Speed is not entirely down to clock speed
  - It’s what you do with it

ARM Conditional Execution

- Often want to conditionally execute some code – while loops, for loops etc

```
cmp al,dl
jg label1
```

  - Traditional approach is to execute a compare
  - Branch if the condition is met
  - Branches are ‘expensive’ (difficult to make parallel efficiently)
- ARM allows any instruction to made conditional
  - Remove the need for an expensive branch
  - Smaller code footprint

ARM and Thumb

- Every ARM instruction is 32 bits long
  - Can take up a lot of memory
    - Poor ”code density”
  - Memory accesses take time
  - Can slow things down
- Thumb
  - 16-bit version of the ARM instruction set
  - Variable length instruction set
  - Took the most popular ARM instructions used and encoded them as 16-bit values

ARM and Thumb

- Speed
  - ARM instructions require 32 bits to be read every instruction
  - Memory reads take time
  - Not all memory is 32 bit wide
    - On smaller RAM width, takes multiple reads to get an instruction
- By making the instructions smaller gain a speed up
  - 8-bit memory two reads per instruction
  - 16-bit memory, one read per instruction
  - 32-bit memory, one read gives two instructions

ARM big.LITTLE

- ARM's latest processor contains two cores
  - 2.3 GHz quad core optimised for performance (big core)
    - High power, high performance
  - 1.7 GHz quad core optimised for energy efficiency (LITTLE core)
    - Low power, lower performance
- Two Cores are architecturally consistent
- System can switch between the two as appropriate for the task in hand

Summary

- Architectural differences between a PC and a mobile phone
  - SoC, PoP, etc.
- Main components of a mobile phone
  - SoC, RF transceiver, modem, sensors, battery, etc.
- Differences between ARM and x86
  - RISC, fixed instruction width, Thumb, big.LITTLE, etc.

Next time

- Introduction to Android development tools
- Android as an operating system
  - What is it?