Edinburgh Napier UNIVERSITY

# RISC : Reduced Instruction Set Computers

Contents:

- ***RISC Vs CISC***
- ***The ARM processors***
- ***Load/Store architectures***
- ***ARM instruction sets***

# The End of CISC?

Edinburgh Napier UNIVERSITY

- Complex Instruction Set Computers (e.g. Pentium) contain large processors running hundreds of possible instructions using dozens of addressing modes.
- Problems include the difficulty of handling variable length instructions, the fact that 80% of instructions are rarely used, and that processors were increasingly limited by external memory speeds (since the original PC, processors have speeded up by X200, memory by X20).
  - (Most of this in Williams, chapter 21)

# The birth of RISC

Edinburgh Napier UNIVERSITY

- During the 80s, an alternative approach arose, and proved to be successful, being widely used in machines such as Sun's SPARC based Unix workstations. Many of the best ideas have been incorporated into other processors (e.g. Pentium).
- The key idea is to keep the instructions simple, but run them fast.
- This has to be coupled with other factors, such as more sophisticated compilers that can convert high level languages into the simple RISC instructions.

# Simple Instructions

Edinburgh Napier UNIVERSITY

- All the same length (usually 4 bytes) and do relatively simple operations. Fancy addressing modes are not supported. They don't need to be microcoded.
- This frees up chip space. This is used for a number of things, in particular lots of identical registers. Register files may be 256 registers long (c.f. the Pentium's 4 general registers). They are identical. Numbers can be kept in the processor, rather than continually being replaced in memory.

# Pipelining

Edinburgh Napier UNIVERSITY

- This is actually a technique from the RISC world. Although I described it in the Pentium, the early members of the family didn't use it. Instructions typically took 4 or more clock cycles to execute.
- The simplicity of RISC instructions make pipelines easy to implement. RISC processors adopted pipelining before CISC processors.
- The SPARC processors were early adopters of multiple pipelines: 'superscalar'.

Edinburgh Napier UNIVERSITY

# A RISC example: ARM

- According to ARM, about 75% of all embedded 32 bit microcontrollers are based on ARM cores.
- They are used in mobile phones (Samsung, Nokia, iPhone), hand-held games machines, iPods, and many other products.
- Typically they are licensed as a core, then incorporated into a manufacturer's design.
- Their popularity is at least partly due to being very power efficient, so they are ideal for battery powered applications.

Edinburgh Napier UNIVERSITY

# ARM families

- There have been a number of families of ARM processors since the line started in the mid 1980s.
- Numbering used to be simple (ARM1 ARM2…)
- Starting with the Cortex cores, the processors have split into three families:
- Application series: Cortex-A
  - Mobile phones & media devices
- Microcontroller series: Cortex-M
  - Simple control applications needing I/O & interrupts
- Real-Time series: Cortex-R
  - Signal processing, floating point & high performance embedded

| CORE | DEVICE | PRODUCT |
| --- | --- | --- |
| Cortex-A8 | Allwinner A10, Allwinner A13, Texas Instruments OMAP3xxx series, Freescale i.MX51-SOC, Freescale i.MX53 QSB, Apple A4, ZiiLABS ZMS-08, Snapdragon, Samsung Hummingbird S5PC100/S5PC110, Marvell ARMADA 500/600, Qualcomm Snapdragon QSD8672/MSM8260/MSM8660(based on Cortex A8), Rockchip RK2918[13] | HTC Desire, SBM7000, Oregon State University OSWALD, Gumstix Overo Earth, Pandora, Apple iPhone 3GS, Apple iPod touch (3rd and 4th Generation), Apple iPad (A4), Apple iPhone 4 (A4), Apple TV (Second Generation) (A4), Archos 5, Archos 43, BeagleBoard, Genesi EFIKA MX, Motorola Droid, Motorola Droid X, Motorola Droid 2, Motorola Droid R2D2 Edition, Palm Pre, Palm Pre 2, HP Veer, HP Pre 3, Samsung Omnia HD, Samsung Wave S8500, Samsung i9000 Galaxy S, Samsung P1000 Galaxy Tab, Sony Ericsson Satio, Sony Ericsson Xperia X10, Touch Book, Nokia N900, Meizu M9, Google Nexus S, Galaxy SL, Sharp PC-Z1 "Netwalker". |
| Cortex-A9 | Texas Instruments OMAP4430/4440, ST-Ericsson NovaThor U8500 / U9500, Nvidia Tegra2, Tegra3, Samsung Orion / Exynos 4210, STMicroelectronics SPEAr1310, Xilinx Extensible Processing Platform,[14] Trident PNX847x/8x/9x STB SoC,[15] Freescale i.MX6,[16] Apple A5 | Samsung Galaxy S II (Samsung Exynos), Sony Xperia U, Samsung Galaxy S III, Apple iPad 2 & iPhone 4S (A5), BlackBerry PlayBook (TI OMAP4430), LG Optimus 2X, LG Optimus 3D, Motorola Atrix 4G, Motorola DROID BIONIC, Motorola Xoom, PandaBoard, PlayStation Vita, HP TouchPad, Acer ICONIA TAB A-series, HTC Sensation, HTC EVO 3D, ASUS Eee Pad Transformer, ASUS Eee Pad Transformer Prime, Lenovo IdeaPad K2 |

Edinburgh Napier UNIVERSITY

# ARM features

- Much smaller, simpler chips than typical Pentium.
- Fast interrupt response for handling hardware.
- Simple RISC instructions (about 50 machine code instructions compared with hundreds for a Pentium).
- All the instructions are the same size and execute (mostly) in one clock.

Edinburgh Napier UNIVERSITY

# Load/Store architecture

- A complex processor like the Pentium has some compound instructions that will do things like getting a variable from memory, performing arithmetic on it and storing the result in a register.
- The ARM processors follow RISC principles: each instruction does one relatively simple instruction. So, for instance, it takes one instruction to get the number into a register, a second to do the arithmetic, and a third to store the result back into memory.

Edinburgh Napier UNIVERSITY

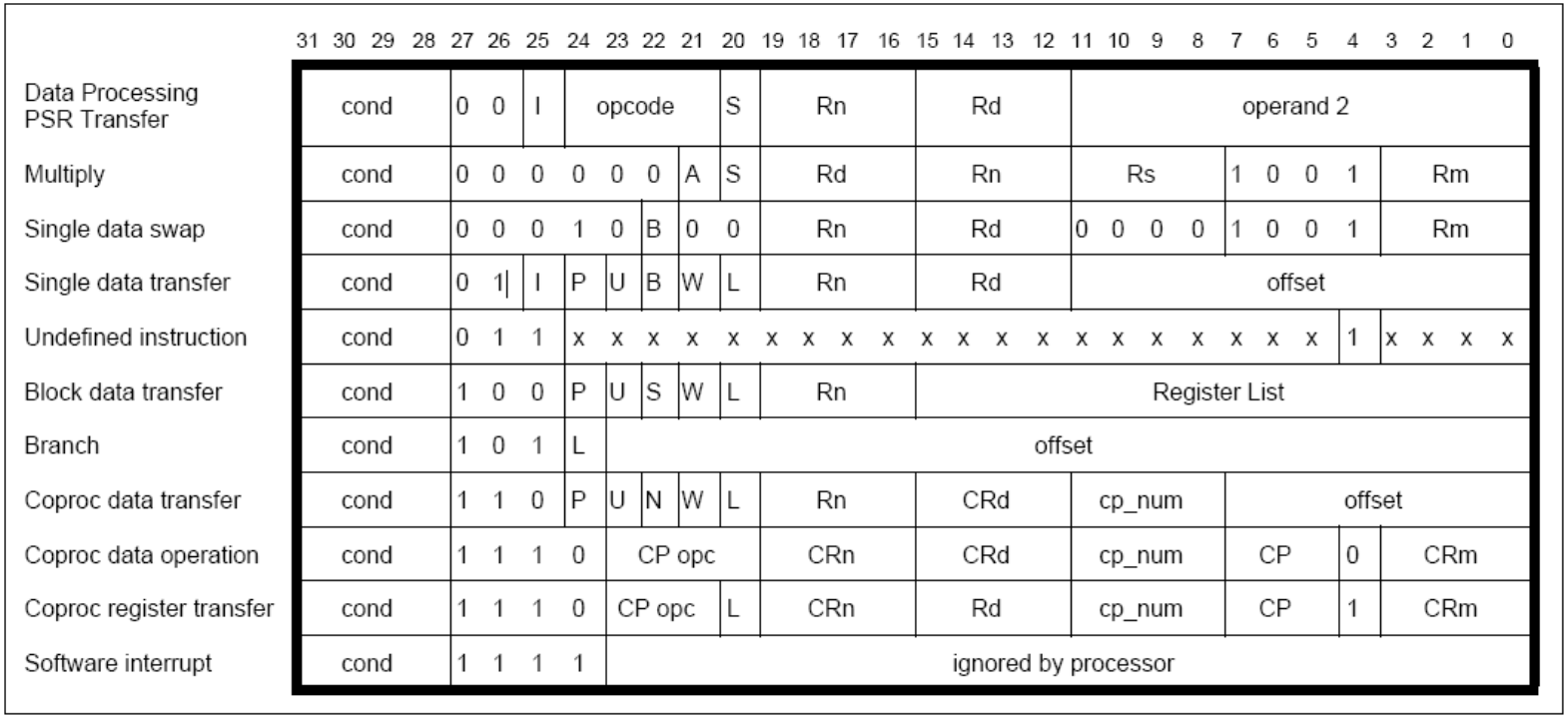
# Example code

```
ldr R1,[R5];    Get the number from the
                memory location pointed to by
                R5 and load it into R1
ldr R2,[R6];    Load R2 from mem[R6]

Add R0,R1,R2;   Add R1+R2, store result in R0
Str R0,[R7];    Store the number in R0 into
                mem[R7] (Note the reversed
                order)
```

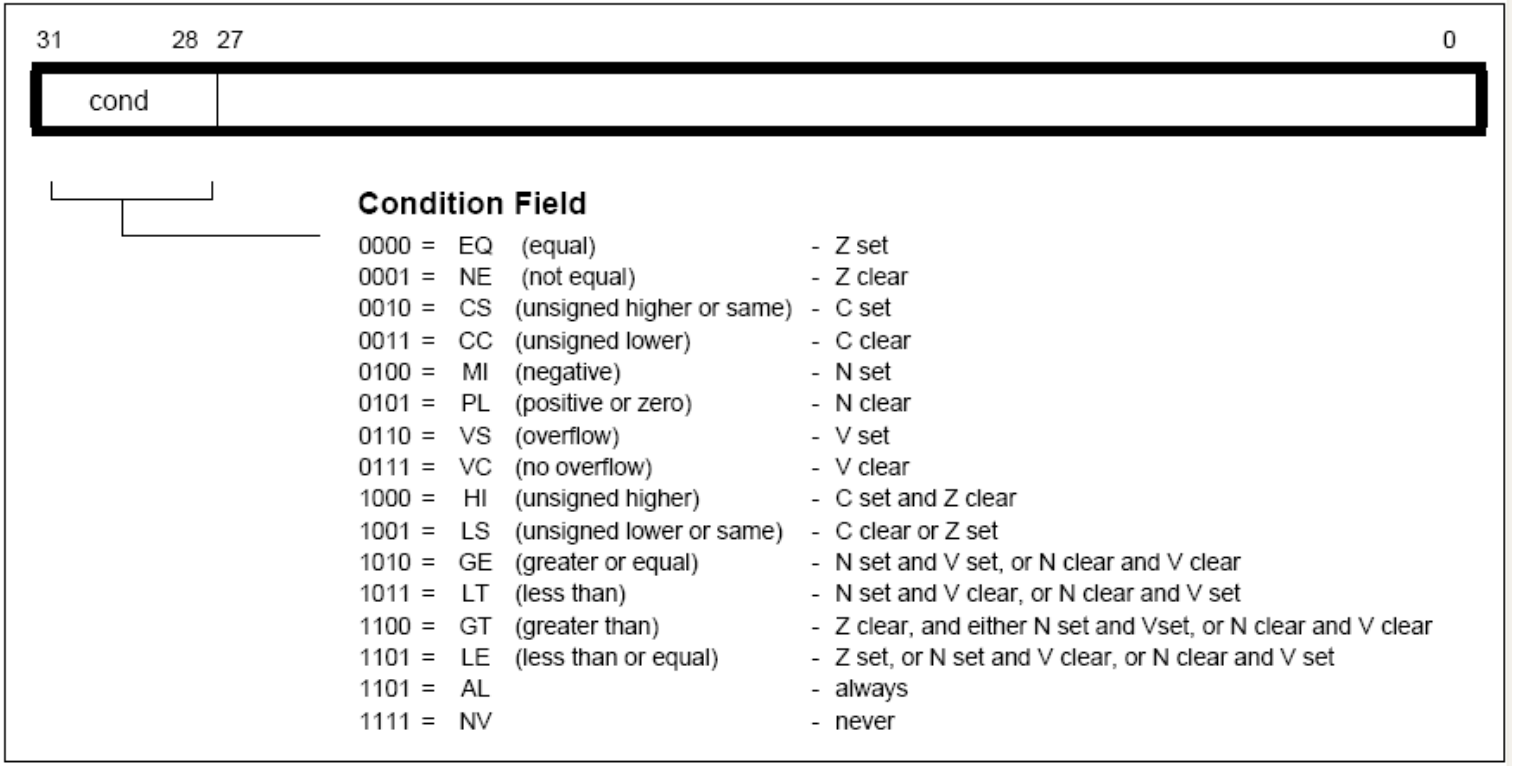
Edinburgh Napier UNIVERSITY

# Summary of RISC instruction set



| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data Processing PSR Transfer | cond | | | | 0 | 0 | I | opcode | | | | S | Rn | | | | Rd | | | | operand 2 | | | | | | | | | | | |
| Multiply | cond | | | | 0 | 0 | 0 | 0 | 0 | 0 | A | S | Rd | | | | Rn | | | | Rs | | | | 1 | 0 | 0 | 1 | Rm | | | |
| Single data swap | cond | | | | 0 | 0 | 0 | 1 | 0 | B | 0 | 0 | Rn | | | | Rd | | | | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | Rm | | | |
| Single data transfer | cond | | | | 0 | 1 | I | P | U | B | W | L | Rn | | | | Rd | | | | offset | | | | | | | | | | | |
| Undefined instruction | cond | | | | 0 | 1 | 1 | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | 1 | x | x | x | x |
| Block data transfer | cond | | | | 1 | 0 | 0 | P | U | S | W | L | Rn | | | | Register List | | | | | | | | | | | | | | | |
| Branch | cond | | | | 1 | 0 | 1 | L | offset | | | | | | | | | | | | | | | | | | | | | | | |
| Coproc data transfer | cond | | | | 1 | 1 | 0 | P | U | N | W | L | Rn | | | | CRd | | | | cp_num | | | | offset | | | | | | | |
| Coproc data operation | cond | | | | 1 | 1 | 1 | 0 | CP opc | | | | CRn | | | | CRd | | | | cp_num | | | | CP | | | 0 | CRm | | |
| Coproc register transfer | cond | | | | 1 | 1 | 1 | 0 | CP opc | | | L | CRn | | | | Rd | | | | cp_num | | | | CP | | | 1 | CRm | | |
| Software interrupt | cond | | | | 1 | 1 | 1 | 1 | ignored by processor | | | | | | | | | | | | | | | | | | | | | | | |

# The Condition Field




**Condition Field**

| | | | |
|---|---|---|---|
| 0000 = | EQ | (equal) | - Z set |
| 0001 = | NE | (not equal) | - Z clear |
| 0010 = | CS | (unsigned higher or same) | - C set |
| 0011 = | CC | (unsigned lower) | - C clear |
| 0100 = | MI | (negative) | - N set |
| 0101 = | PL | (positive or zero) | - N clear |
| 0110 = | VS | (overflow) | - V set |
| 0111 = | VC | (no overflow) | - V clear |
| 1000 = | HI | (unsigned higher) | - C set and Z clear |
| 1001 = | LS | (unsigned lower or same) | - C clear or Z set |
| 1010 = | GE | (greater or equal) | - N set and V set, or N clear and V clear |
| 1011 = | LT | (less than) | - N set and V clear, or N clear and V set |
| 1100 = | GT | (greater than) | - Z clear, and either N set and Vset, or N clear and V clear |
| 1101 = | LE | (less than or equal) | - Z set, or N set and V clear, or N clear and V set |
| 1101 = | AL | | - always |
| 1111 = | NV | | - never |

Edinburgh Napier UNIVERSITY

# Instruction types:- Thumb

- Having fixed-length instructions makes memory fetches much simpler to implement. However, the standard ARM 32-bit instruction is a bit wasteful of storage.
- Many ARM processors can now use the Thumb instructions. These are the commonest instructions compressed down to 16 bits.
- They can be used in small systems with a reduced 16-bit data bus for low cost.

Edinburgh Napier UNIVERSITY

# Instruction Types: Jazelle

- Some recent ARM processors use this technology to execute Java bytecode (the intermediate code produced by Java compilers).
- Most bytecode instructions can be executed directly in hardware (some need some software).
- This is used in mobile phones for running Java applications and games.
- Question: How is Java executed in a traditional Pentium-based PC?

Java bytecode needs to be converted by the Java Runtime Environment into the machine code of the processor it is being run on.

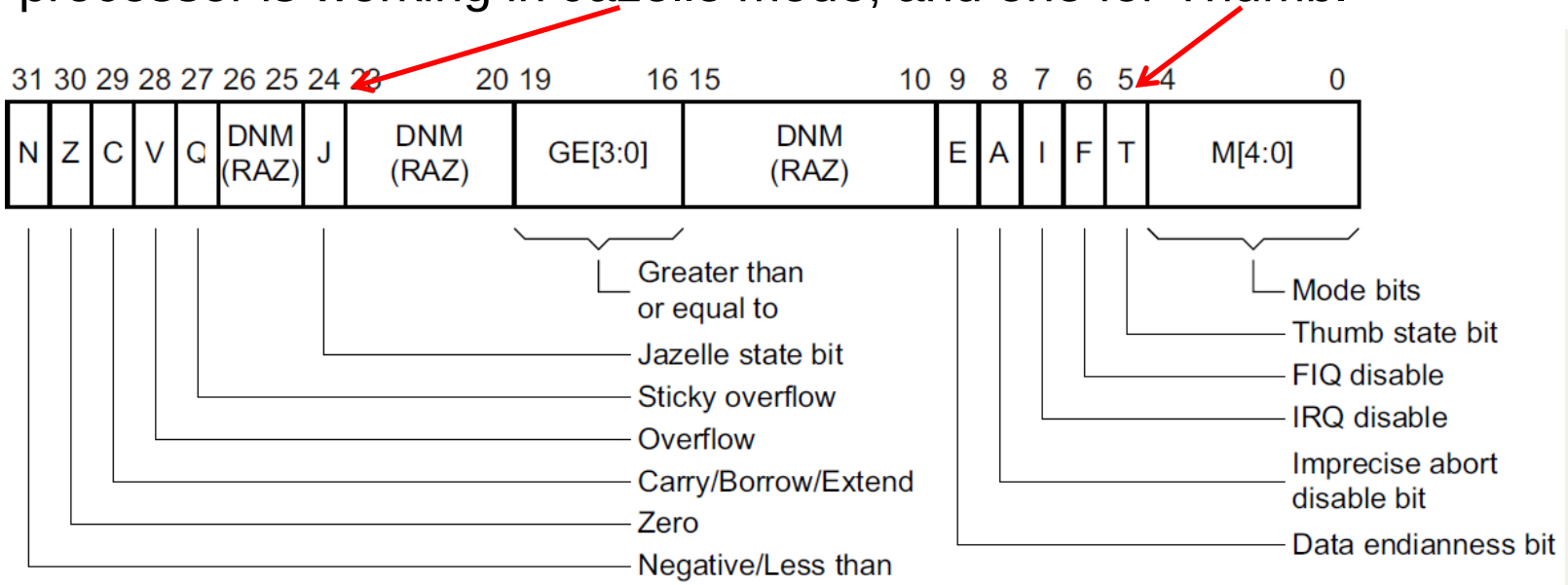
# Registers

- There are 16 main registers (actually not many for a RISC design). All are 32 bits.
- R0 – R12 are available for general use. R13-R15 have special purposes.
- For instance R13 is the Stack Pointer and R15 is the program counter.

| | |
|---|---|
| R0 | R1 |
| R2 | R3 |
| R4 | R5 |
| R6 | R7 |
| R8 | R9 |
| R10 | R11 |
| R12 | R13/sp |
| R14/lr | R15/pc |

# Flags register

- On the ARM, this is called the Current Program Status Register (CPSR). It is a 32 bit register, but not all the bits are used.
- Some of the bits are used for traditional flags (zero, overflow, negative, interrupt enabled).
- Some control the mode of working. So, there is one to control if the processor is working in Jazelle mode, and one for Thumb.



# Main ARM variants

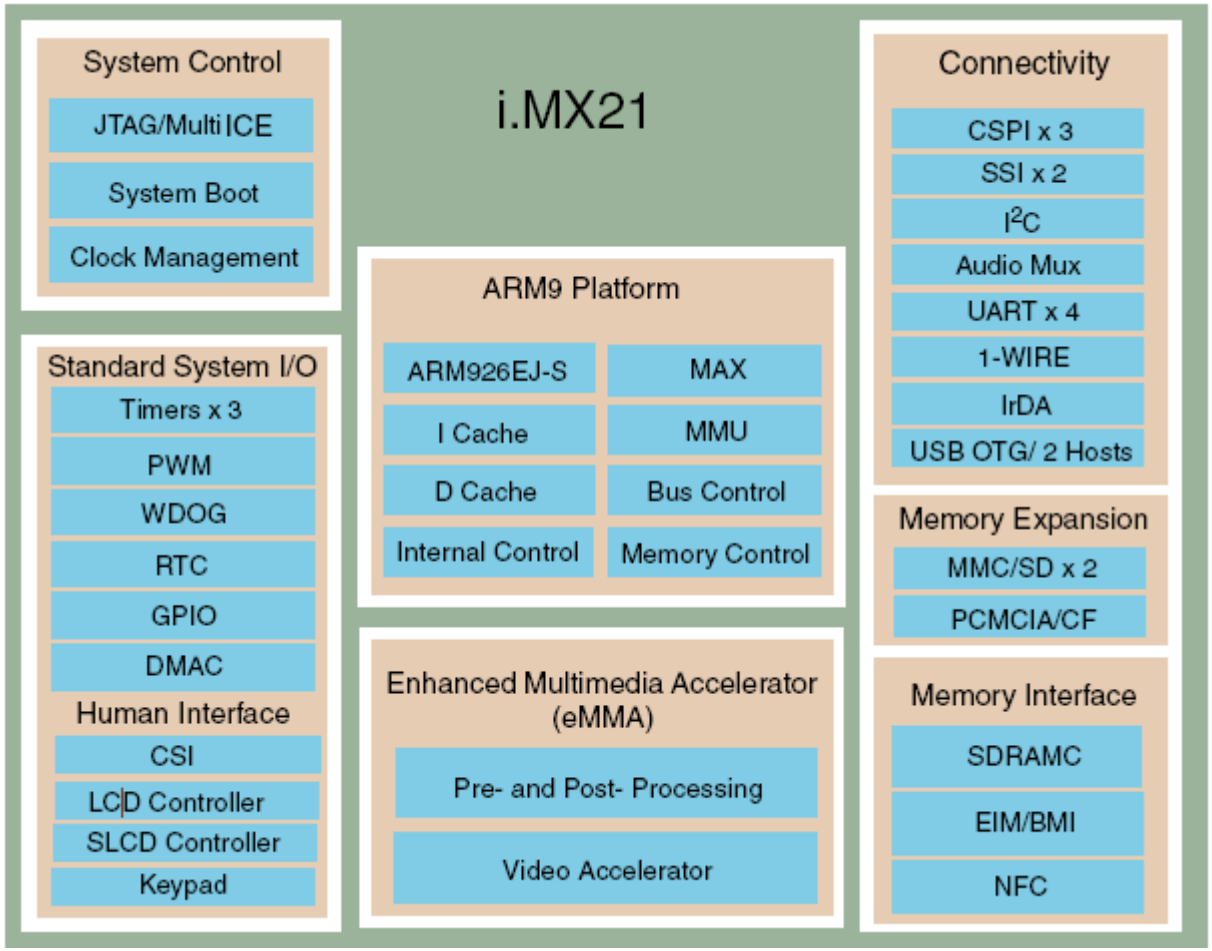
Edinburgh Napier UNIVERSITY

| **Application** Processors | Embedded Processors | SecurCores |
|---|---|---|
| ARM Cortex-A8 | ARM Cortex-M1 | SecurCore SC100 |
| ARM Cortex-A9 MPCore | ARM Cortex-M3 | SecurCore SC200 |
| ARM Cortex-A9 Single Core Processor | ARM Cortex-R4(F) | |
| ARM11 MPCore | ARM1156T2(F)-S | |
| ARM1136J(F)-S | ARM7EJ-S | |
| ARM1176JZ(F)-S | ARM7TDMI | |
| ARM720T | ARM7TDMI-S | |
| ARM920T | ARM946E-S | |
| ARM922T | ARM966E-S | |
| ARM926EJ-S | ARM968E-S | |
| | ARM996HS | |

Manufacturers

Edinburgh Napier UNIVERSITY

# Freescale Semiconductor's MC9328MX21




Note how the ARM is only the core, with application specific I/O added to the chip.