ARM Processor - Architecture

Lecture on ARM7 Architecture

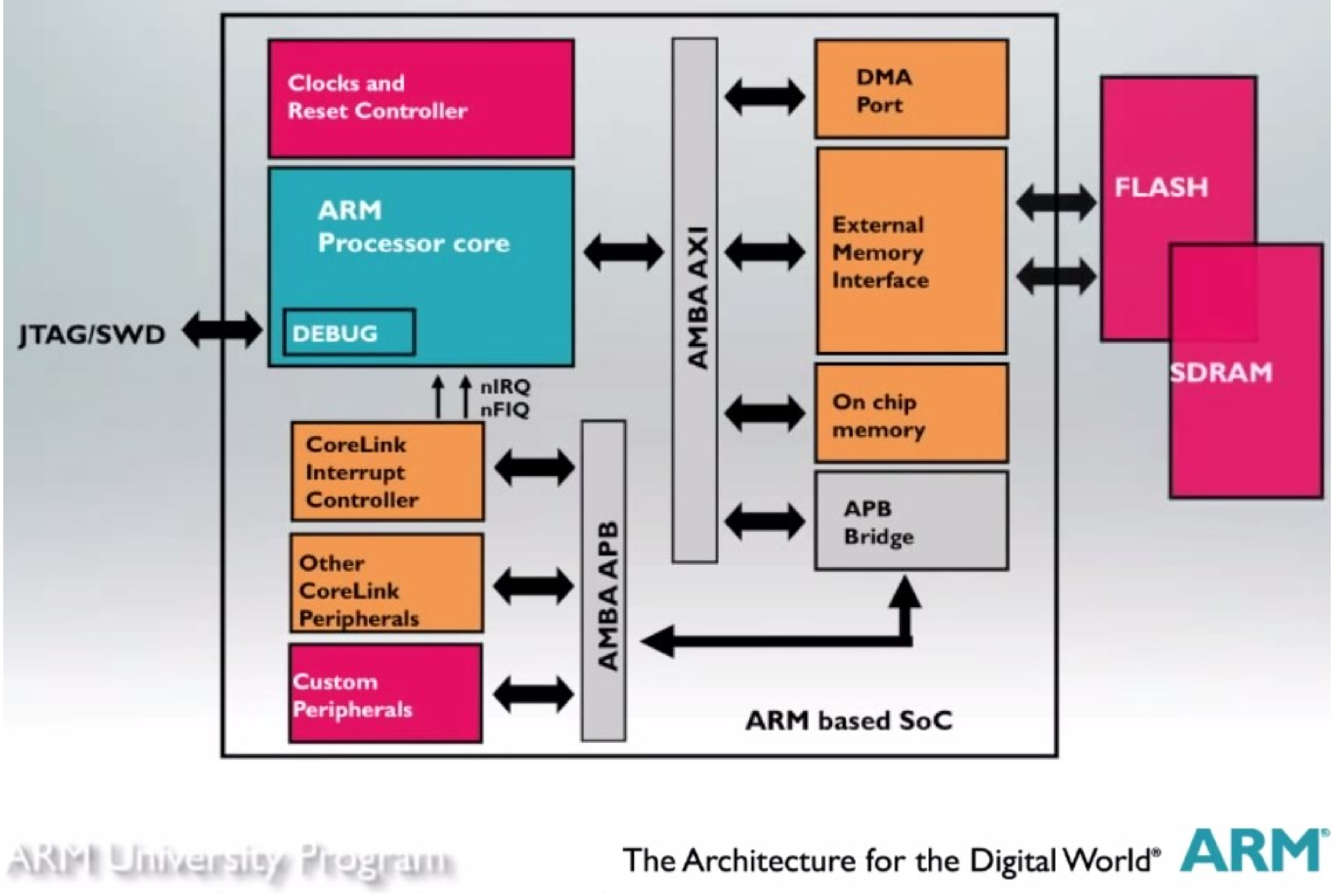
By

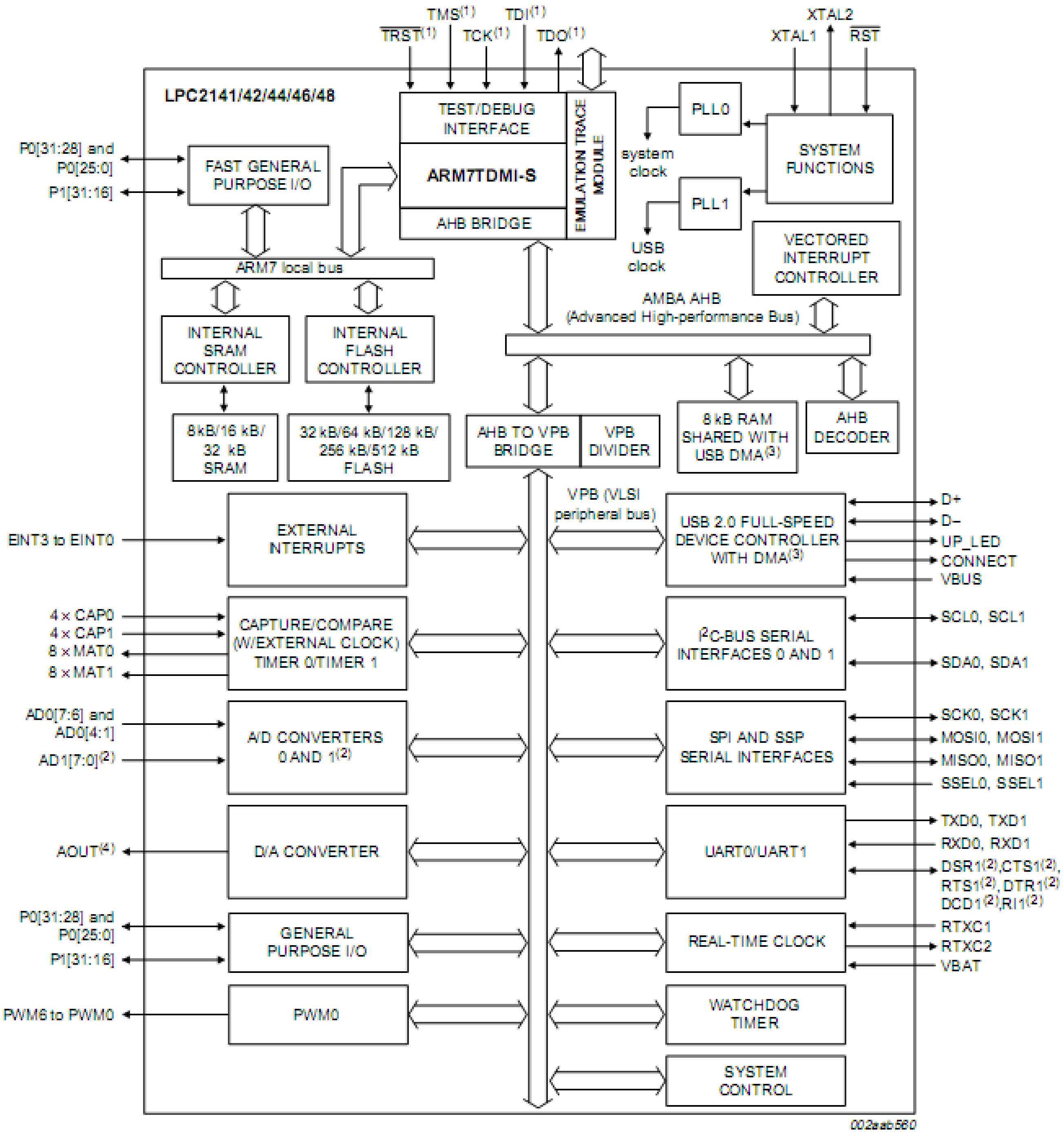
Harish V. Mekali

Assistant Professor, Dept. of ECE

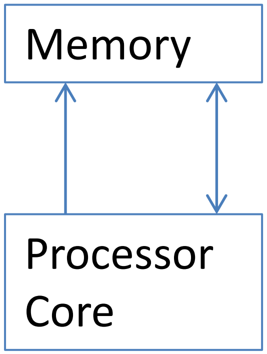
BMSCE, Bangalore - 19

Inside ARM based System

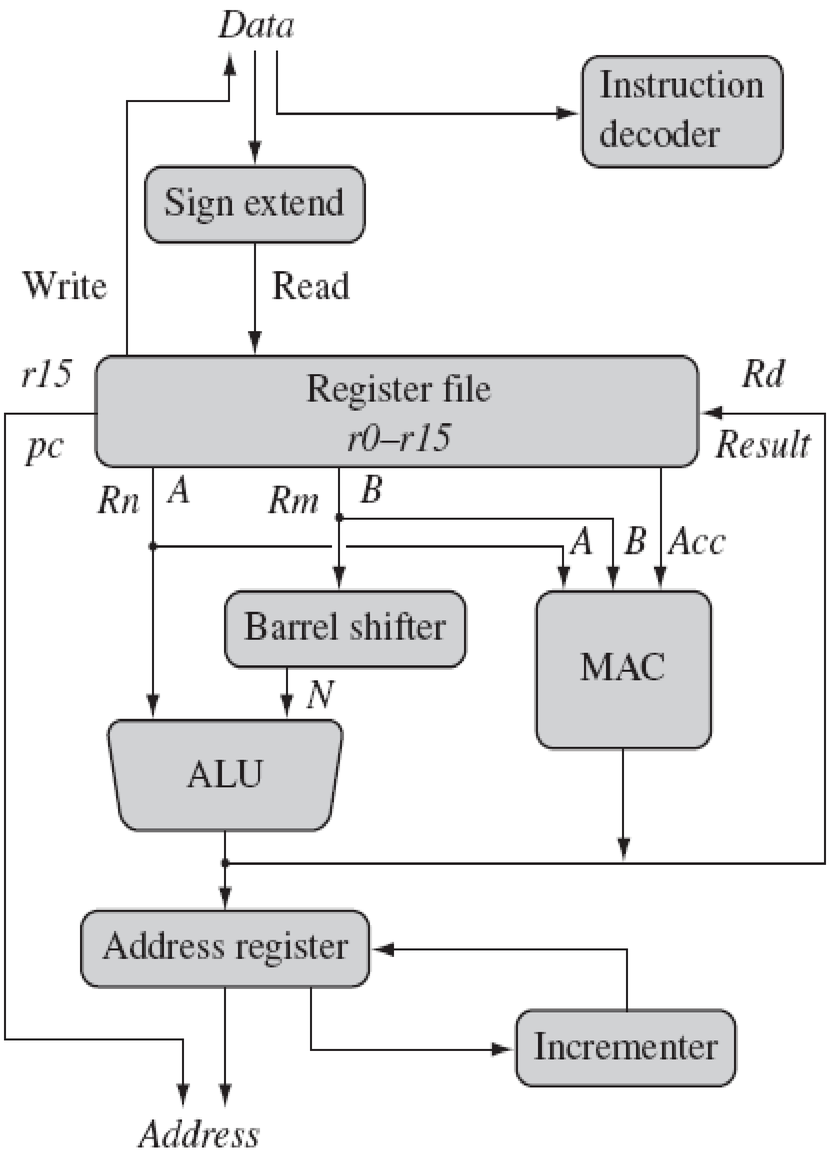


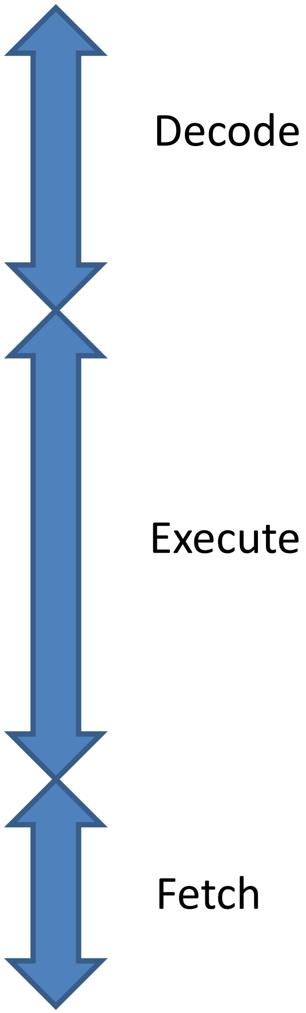

LPC2141/42/44/46/48
TMS(1)
TDI(1)
TRST(1)
TCK(1)
TDO(1)
XTAL2
XTAL1
RST
TEST/DEBUG INTERFACE
ARM7TDMI-S
AHB BRIDGE
EMULATION TRACE MODULE
PLL0
PLL1
system clock
USB clock
SYSTEM FUNCTIONS
VECTORED INTERRUPT CONTROLLER
P0[31:28] and P0[25:0]
P1[31:16]
FAST GENERAL PURPOSE I/O
ARM7 local bus
INTERNAL SRAM CONTROLLER
INTERNAL FLASH CONTROLLER
8 kB/16 kB/ 32 kB SRAM
32 kB/64 kB/128 kB/ 256 kB/512 kB FLASH
AMBA AHB (Advanced High-performance Bus)
AHB TO VPB BRIDGE
VPB DIVIDER
8 kB RAM SHARED WITH USB DMA(3)
AHB DECODER
VPB (VLSI peripheral bus)
EINT3 to EINT0
EXTERNAL INTERRUPTS
USB 2.0 FULL-SPEED DEVICE CONTROLLER WITH DMA(3)
D+
D−
UP_LED
CONNECT
VBUS
4 × CAP0
4 × CAP1
8 × MAT0
8 × MAT1
CAPTURE/COMPARE (W/EXTERNAL CLOCK) TIMER 0/TIMER 1
I2C-BUS SERIAL INTERFACES 0 AND 1
SCL0, SCL1
SDA0, SDA1
AD0[7:6] and AD0[4:1]
AD1[7:0](2)
A/D CONVERTERS 0 AND 1(2)
SPI AND SSP SERIAL INTERFACES
SCK0, SCK1
MOSI0, MOSI1
MISO0, MISO1
SSEL0, SSEL1
AOUT(4)
D/A CONVERTER
UART0/UART1
TXD0, TXD1
RXD0, RXD1
DSR1(2),CTS1(2), RTS1(2), DTR1(2) DCD1(2),RI1(2)
P0[31:28] and P0[25:0]
P1[31:16]
GENERAL PURPOSE I/O
REAL-TIME CLOCK
RTXC1
RTXC2
VBAT
PWM6 to PWM0
PWM0
WATCHDOG TIMER
SYSTEM CONTROL
002aab560

Programs model / Data flow Model



Ex instruction :
ADD Rd, Rn, Rm, LSL #2
i.e., Rd = Rn + Rm * 4





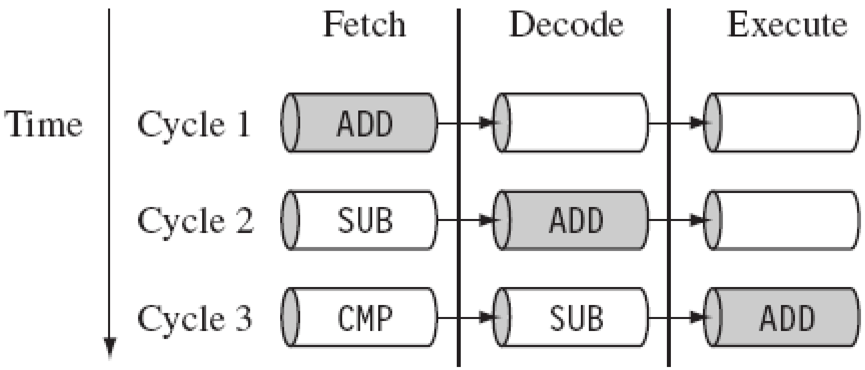
Pipelining

Ex program in ASM :

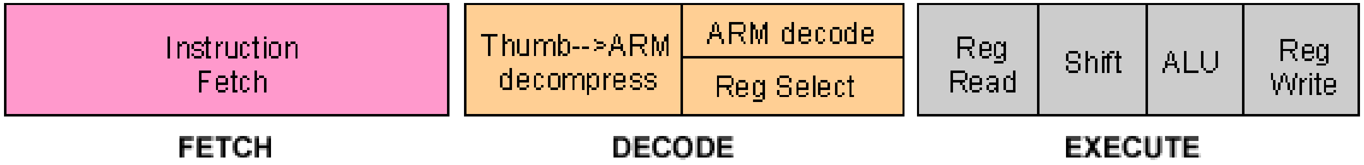
ADD

SUB

CMP



ARM7TDMI - Pipeline



Processors States

- When the processor is executing in ARM state:
  - All instructions are 32 bits wide
  - All instructions must be word aligned
  - Therefore the pc value is stored in bits [31:2] with bits [1:0] undefined (as instruction cannot be halfword or byte aligned).

- When the processor is executing in Thumb state:
  - All instructions are 16 bits wide
  - All instructions must be halfword aligned
  - Therefore the pc value is stored in bits [31:1] with bit [0] undefined (as instruction cannot be byte aligned).

- When the processor is executing in Jazelle state:
  - All instructions are 8 bits wide
  - Processor performs a word access to read 4 instructions at once

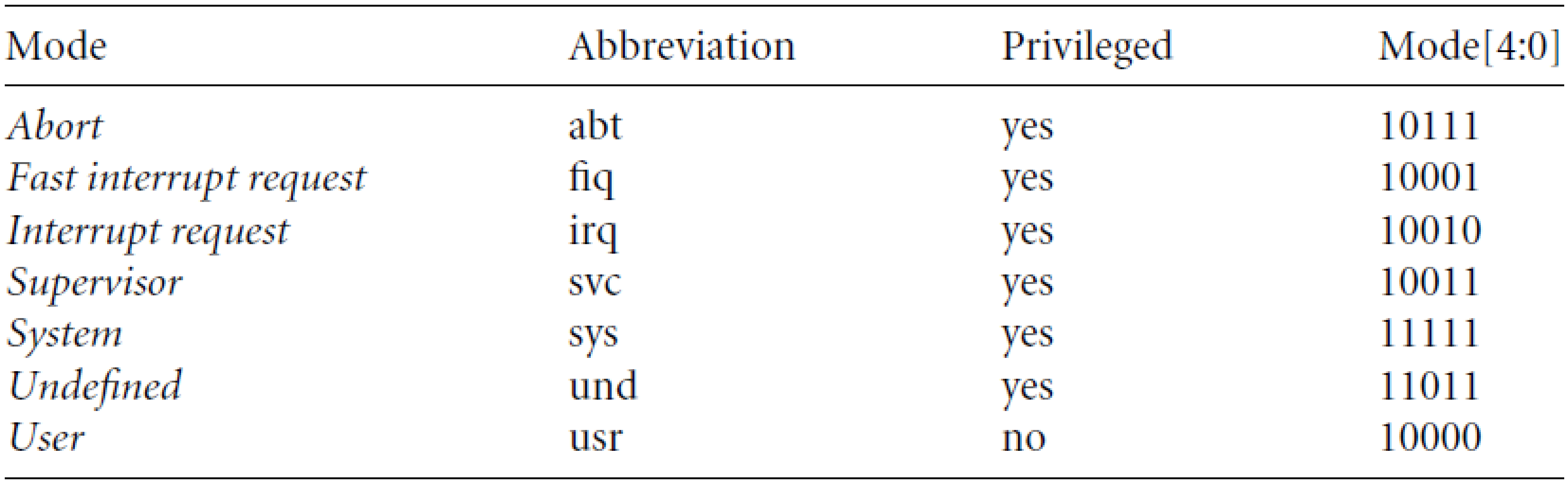
- Hence there are 3 instruction sets , We can not intermingle the instructions and for Jazelle state T=0 and J=1 in CPSR

Processors Modes

- **ARM has seven basic operating modes**
  - Each mode has access to its own stack space and a different subset of registers
  - Some operations can only be carried out in a privileged mode

	Mode	Description	
Exception modes	**Supervisor (SVC)**	Entered on reset and when a Supervisor call instruction (SVC) is executed	**Privileged modes**
	FIQ	Entered when a high priority (fast) interrupt is raised	
	IRQ	Entered when a normal priority interrupt is raised	
	Abort	Used to handle memory access violations	
	Undef	Used to handle undefined instructions	
	System	Privileged mode using the same registers as User mode	
	User	Mode under which most Applications / OS tasks run	**Unprivileged mode**

Processors Modes



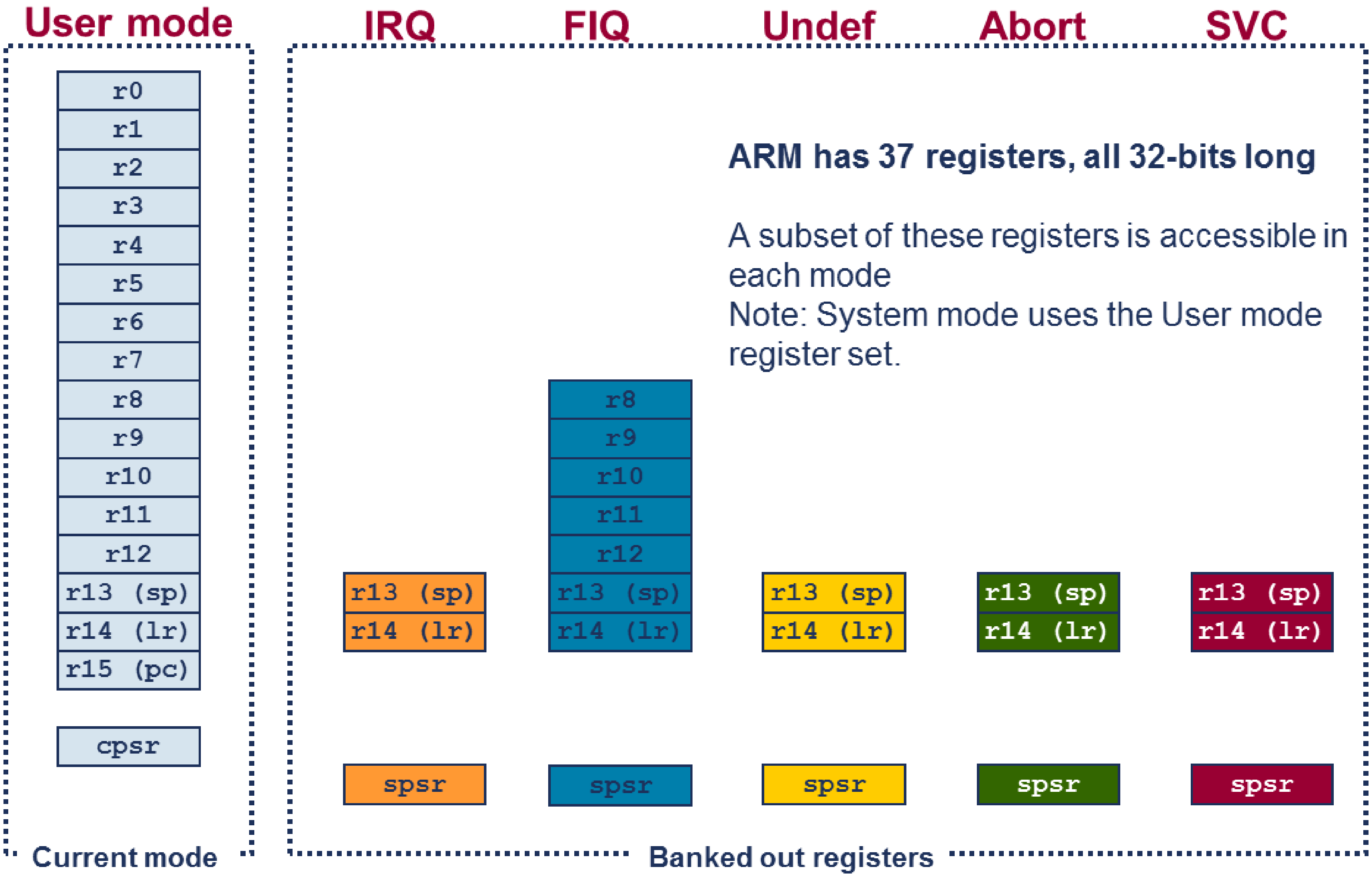
Mode	Abbreviation	Privileged	Mode[4:0]
Abort	abt	yes	10111
Fast interrupt request	fiq	yes	10001
Interrupt request	irq	yes	10010
Supervisor	svc	yes	10011
System	sys	yes	11111
Undefined	und	yes	11011
User	usr	no	10000

Non privileged modes allows read access to **control field** in CPSR but, allows read – write access to **conditional flag**

Register Organization

ARM has 37 registers, all 32-bits long

A subset of these registers is accessible in each mode
Note: System mode uses the User mode register set.



User mode	IRQ	FIQ	Undef	Abort	SVC
r0					
r1					
r2					
r3					
r4					
r5					
r6					
r7					
r8		r8			
r9		r9			
r10		r10			
r11		r11			
r12		r12			
r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)
r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)
r15 (pc)					
cpsr	spsr	spsr	spsr	spsr	spsr

Register Organization

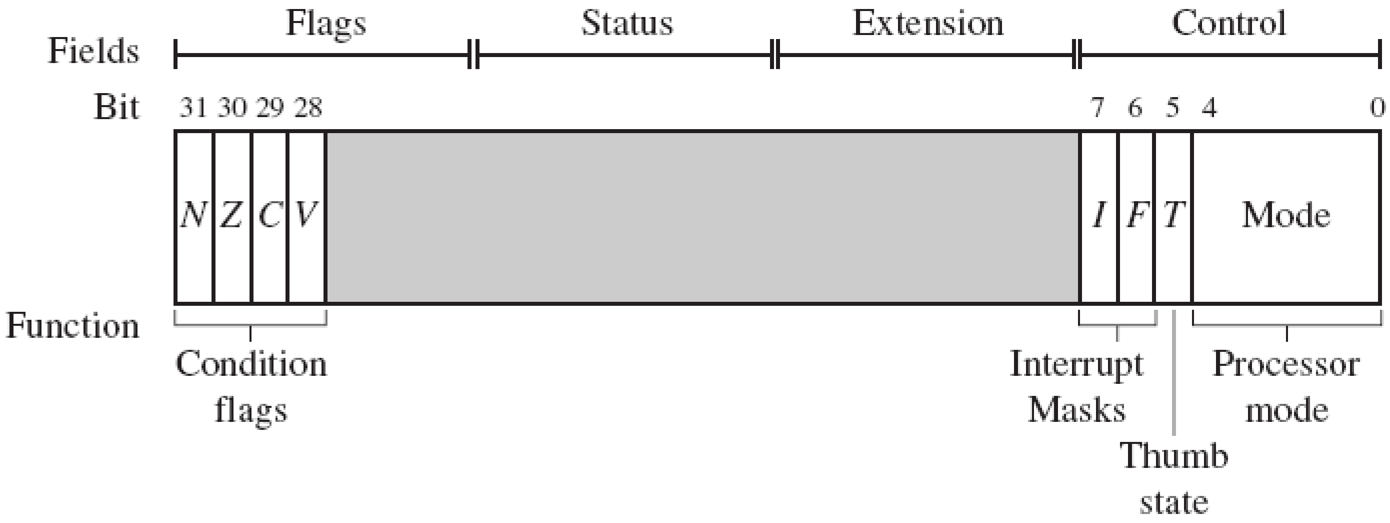
- ARM has 37 registers all of which are 32-bits long.
  - 1 dedicated program counter
  - 1 dedicated current program status register
  - 5 dedicated saved program status registers
  - 30 general purpose registers

- The current processor mode governs which of several banks is accessible. Each mode can access
  - a particular set of r0-r12 registers
  - a particular r13 (the stack pointer, sp) and r14 (the link register, lr)
  - the program counter, r15 (pc)
  - the current program status register, cpsr

- Registers from **r0** to **r13** are orthogonal : Any instruction that we can apply on **r0** can also be used on any other register

- Depending on the context , r13 and r14 can also be used as a general purpose registers

Register Organization - CPSR

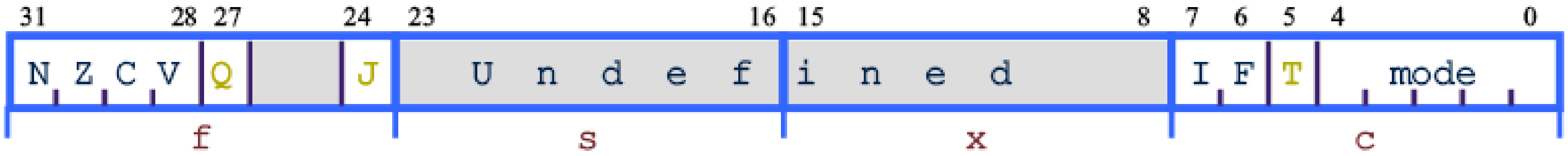


- Condition flags updated by comparisons and ALU operations that specify the “**S**” instruction suffix. **ex : SUBS r0, r1, r2**

- Jazelle enabled cores have **J** bit in CPSR 24th bit. Jazelle state requires a separate software from ARM Ltd., and Sun microsystems.

- Most ARM instructions can be executed conditionally on the value of condition **ex : ADDEQ r0, r1, r2**

Register Organization - CPSR

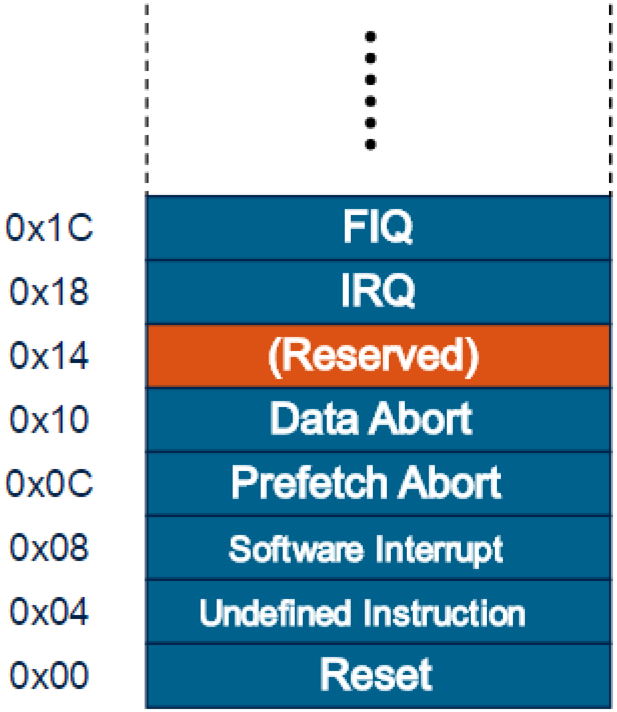


- **Condition code flags**
  - N = Negative result from ALU
  - Z = Zero result from ALU
  - C = ALU operation Carried out
  - V = ALU operation oVerflowed
- **Sticky Overflow flag - Q flag**
  - Architecture 5TE/J only
  - Indicates if saturation has occurred
- **J bit**
  - Architecture 5TEJ only
  - J = 1: Processor in Jazelle state
- **Interrupt Disable bits.**
  - I = 1: Disables the IRQ.
  - F = 1: Disables the FIQ.
- **T Bit**
  - Architecture xT only
  - T = 0: Processor in ARM state
  - T = 1: Processor in Thumb state
- **Mode bits**
  - Specify the processor mode

Exception Handling

- When an exception occurs, the ARM:
  - Copies CPSR into SPSR_<mode>
  - Sets appropriate CPSR bits
    - Change to ARM state
    - Change to exception mode
    - Disable interrupts (if appropriate)
  - Stores the return address in LR_<mode>
  - Sets PC to vector address
- To return, exception handler needs to:
  - Restore CPSR from SPSR_<mode>
  - Restore PC from LR_<mode>

This can only be done in ARM state.



0x1C	FIQ
0x18	IRQ
0x14	(Reserved)
0x10	Data Abort
0x0C	Prefetch Abort
0x08	Software Interrupt
0x04	Undefined Instruction
0x00	Reset

Vector Table

Vector table can be at
0xFFFF0000 on ARM720T
and on ARM9/10 family devices

Core Extensions

- Hardware extensions that can be used to improve the system performance like

1. Cache
2. Memory Management
3. Co-processor

Cache Memory

- Cache is block of fast memory placed between main memory and the core
- With cache processor core run faster majority of the time by without having to wait for the data from slow external memory
- Cache can be of Harvard and Von neumann style
- Cache provides Overall increase in performance at the expense of **predictable logic / execution**

Core Extensions

- Most of the ARM based systems uses Single level cache

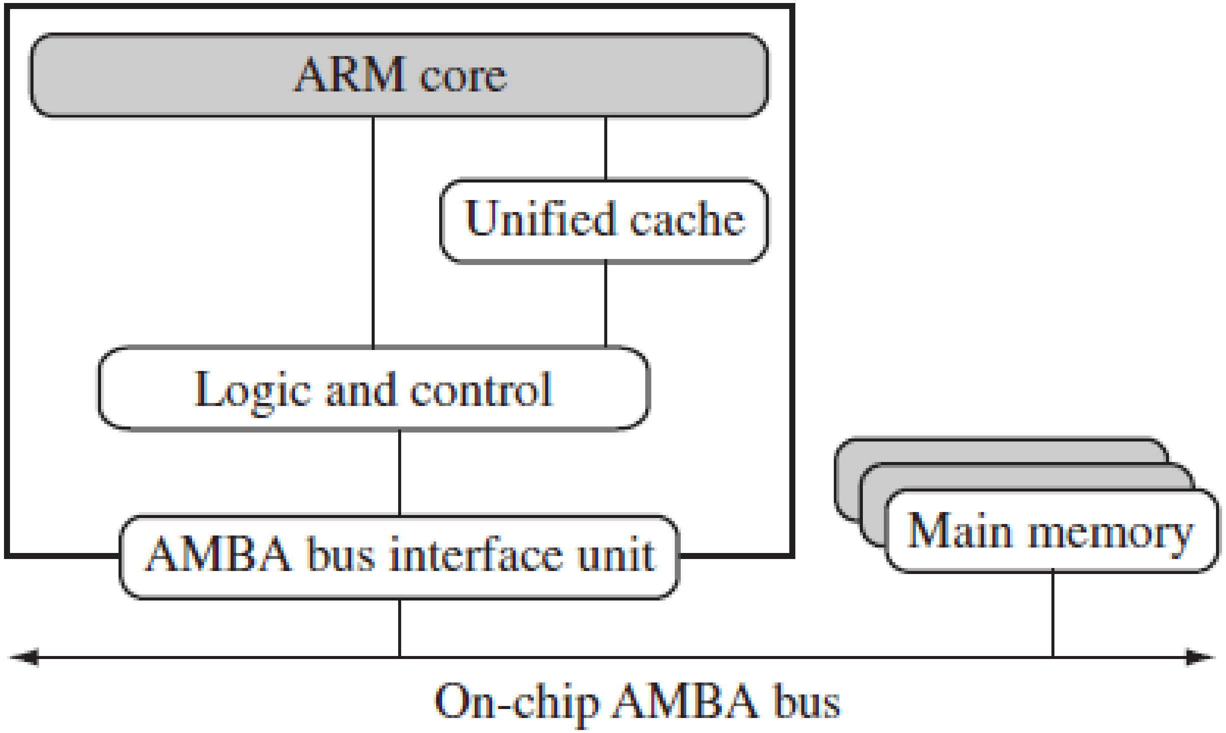


Fig : Simplified von Neumann architecture of cache

Core Extensions

Memory Management

- Embedded Systems have multiple memory devices and it is necessary to organize these devices and protect system from applications trying to make inappropriate access to the hardware
- There are 3 types of memory management hardwares

  1. No extension – No protection

     For very simple embedded applications

  2. Memory Protection Unit(MPU) – Limited Protection

     Each memory region is protected with access control

  3. Memory Management Unit(MMU) – Full protection

     Translation table from virtual to physical memory address as well as permission to access

Core Extensions

Co – Processor

- It extends the processing feature of the basic processor core by extending the instructions or by providing the configuration registers
- They can be accessed through the group of dedicated ARM instructions and it is identified during the decoding stage.
- If in case the coprocessor is not present then basic core generates undefined exception that allows to emulate the coprocessor behaviour in software
- Ex : Vector floating point co-processor

Thank you

Harish V. Mekali
Assistant Professor, Dept. of ECE, BMSCE

hvm.ece@bmsce.ac.in / harishmekali@gmail.com

+91-9538765141

http://harishvmekali.blogspot.in/p/technology.html

I acknowledge and appreciate ARM University Program(AUP) and ARM Embedded Systems pvt. Ltd. for their continuous support.

References

Video lectures :
1. Mr. Chrish Shore, ARM Training Manager, UK
The ARM University Program, ARM Architecture Fundamentals
(https://www.youtube.com/watch?v=7LqPJGnBPMM)
2. Dr. Santanu Chaudhury, Dept. of Electrical Engineering, IIT Delhi
Lecture - 5 ARM : (https://www.youtube.com/watch?v=4VRtujwa_b8)

Website :
3. http://infocenter.arm.com/help/index.jsp

Textbooks:
1. ARM system developers guide, Andrew N Sloss, Dominic Symes and Chris Wright, Elsevier, Morgan Kaufman publishers, 2008.
2. ARM System-on-Chip Architecture, Steve Furber, Second Edition, Pearson, 2015