YCCE, Nagpur

MGI

Embedded System
ARM

Dr. T.G.Panse
Assistant Professor
Department of Electronics Engineering
Yeshwantrao Chavan College of Engineering, Nagpur (M. S.)
Email: 9061@ycce.in
Website: www.ycce.edu

V SEMESTER
23VLS1501 : Embedded System

Unit 1: Introduction to ARM, Advantages of architectural features of ARM Processor, Processor modes, Register organization, Exceptions and its handling, 3/5- stage pipeline ARM organization.

Unit 2 : ARM and THUMB instruction sets, ARM programmer's model, addressing modes,Instruction set in detail and programming, data processing instruction, data transfer instruction, Control flow instructions, simple assembly language programs.

Unit 3 : ARM assembly language programs and C language programs. Code conversion programs.

Unit 4: LPC 2148 architecture block diagrams, pins and signals. GPIO, I / O Interfaces like LED and Switch and their Programs.

Unit 5: Display interfacing with LPC 2148. 7segment display interfacing. LCD interfacing and programs.

Unit 6: LPC 2148 TIMER and PWM Applications. Embedded ARM applications

Text books

1 ARM System on-chip Architecture, 2nd edition, 2000, Steve Furber, Pearson Education Asia

2 Embedded Linux, Hardware, Software and interfacing, 2002. Craig Hallabaugh, Addison-Wesley Professional

3 ARM System Developer's Guide: Designing and Optimizing, 2005 Sloss Andrew N, Symes Dominic, Wright Chris Morgan Kaufman Publication

Introduction

ARM history

The ARM microcontroller stands for **Advance RISC Machine**; it is one of the extensive and most licensed processor cores in the world.

The first ARM processor was developed in the year 1978 by Cambridge University, and the first ARM RISC processor was produced by the Acorn Group of Computers in the year 1985.

These processors are specifically **used in portable devices like digital cameras, mobile phones**, home networking modules and wireless communication technologies and other embedded systems due to the benefits, such as low power consumption, reasonable performance, etc.

ARM Architecture

The ARM architecture processor is an advanced reduced instruction set computing [RISC] machine and it's a 32bit reduced instruction set computer (RISC) microcontroller.

It was introduced by the Acron computer organization in 1987.

This ARM is a family of microcontroller developed by makers like ST Microelectronics, Motorola, Texas and Philips etc.

The ARM architecture comes with totally different versions like ARMv1, ARMv2, etc., and, each one has its own advantage and disadvantages.

Why ARM?

- One of the most licensed and thus widespread processor cores in the world
  - Used in PDA, cell phones, multimedia players, handheld game console, digital TV and cameras
  - ARM7: GBA, iPod
  - ARM9: NDS, PSP, Sony Ericsson, BenQ
  - ARM11: Apple iPhone, Nokia N93, N800
  - 90% of 32-bit embedded RISC processors are ARM
- Used especially in portable devices due to its low power consumption and reasonable performance

ARM powered products

Popular ARM architectures

- ARM7TDMI
  - 3 pipeline stages (fetch/decode/execute)
  - High code density/low power consumption
  - One of the most used ARM-version (for low-end systems)
  - All ARM cores after ARM7TDMI include TDMI even if they do not include TDMI in their labels
- ARM9TDMI
  - Compatible with ARM7
  - 5 stages (fetch/decode/execute/memory/write)
  - Separate instruction and data cache
- ARM11

ARM family comparison

ARM family attribute comparison.

	ARM7	ARM9	ARM10	ARM11
Pipeline depth	three-stage	five-stage	six-stage	eight-stage
Typical MHz	80	150	260	335
mW/MHz[a]	0.06 mW/MHz	0.19 mW/MHz (+ cache)	0.5 mW/MHz (+ cache)	0.4 mW/MHz (+ cache)
MIPS[b]/MHz	0.97	1.1	1.3	1.2
Architecture	Von Neumann	Harvard	Harvard	Harvard
Multiplier	8 × 32	8 × 32	16 × 32	16 × 32

[a] Watts/MHz on the same 0.13 micron process.

[b] MIPS are Dhrystone VAX MIPS.

ARM is a RISC

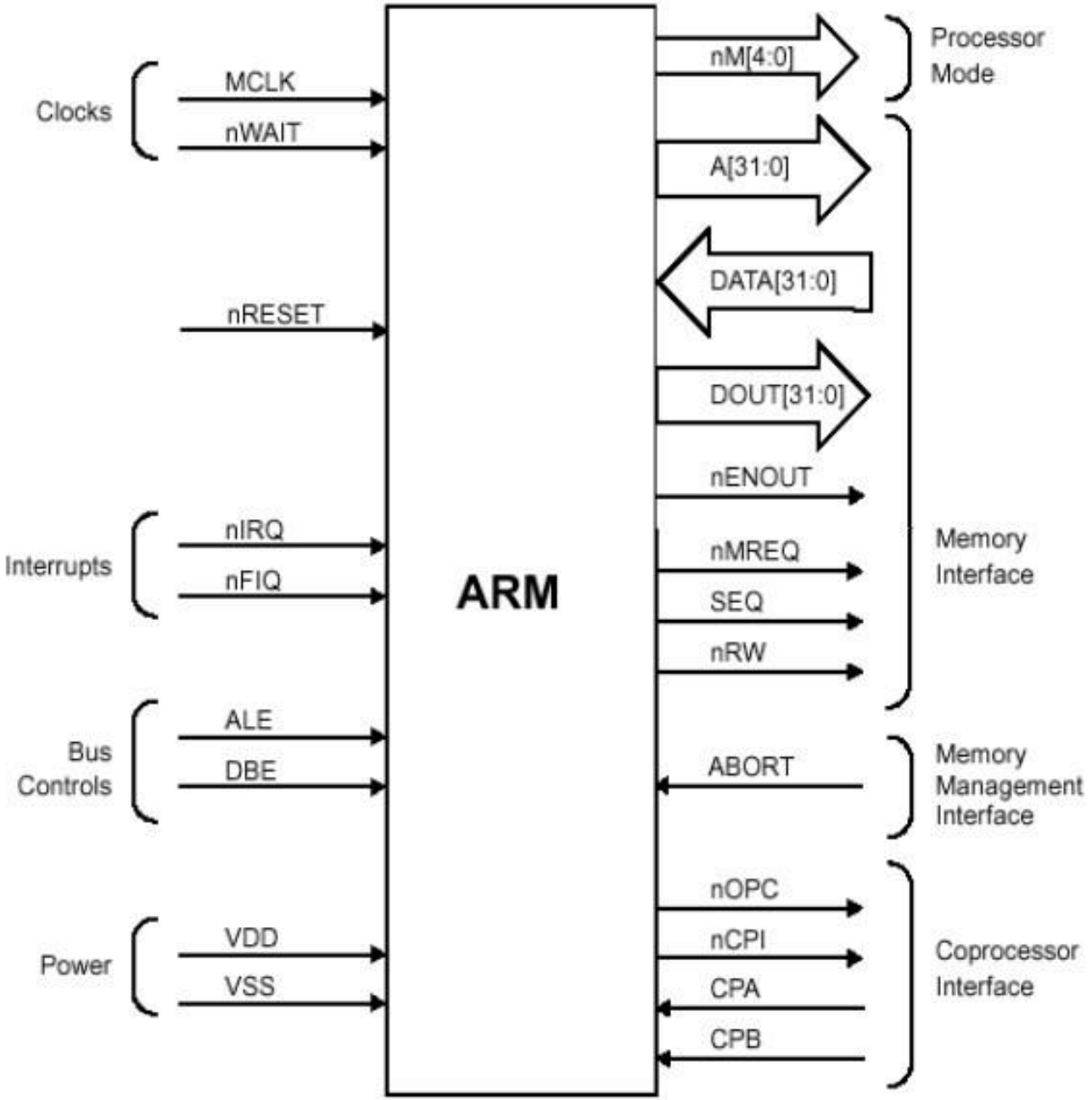
- RISC: simple but powerful instructions that execute within a single cycle at high clock speed.
- Four major design rules:
  - Instructions: reduced set/single cycle/fixed length
  - Pipeline: decode in one stage/no need for microcode
  - Registers: a large set of general-purpose registers
  - Load/store architecture: data processing instructions apply to registers only; load/store to transfer data from memory
- Results in simple design and fast clock rate

ARM design philosophy

- Small processor for lower power consumption (for embedded system)
- High code density for limited memory and physical size restrictions
- The ability to use slow and low-cost memory
- Reduced die size for reducing manufacture cost and accommodating more peripherals

ARM Architecture

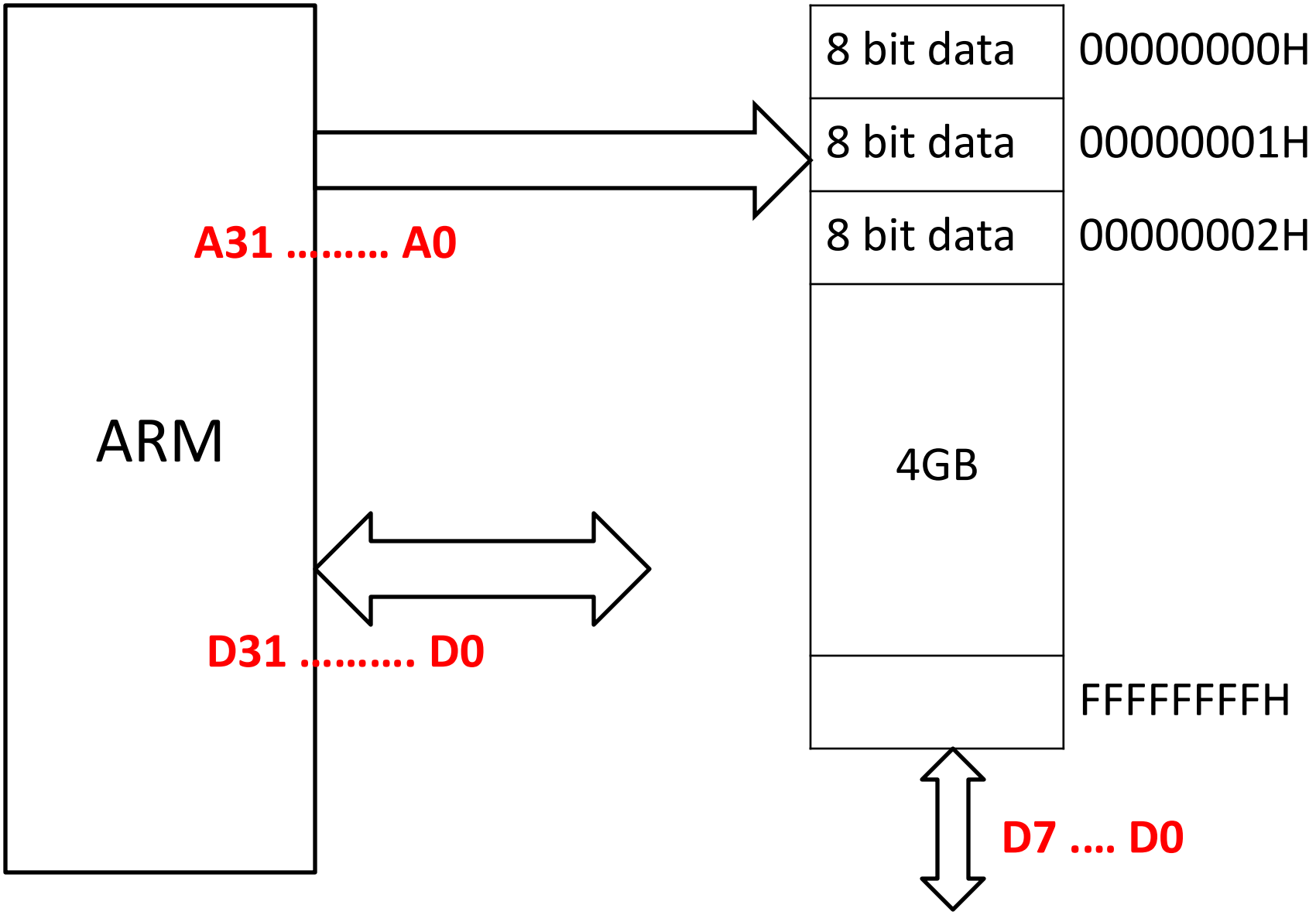
ARM Functional Diagram

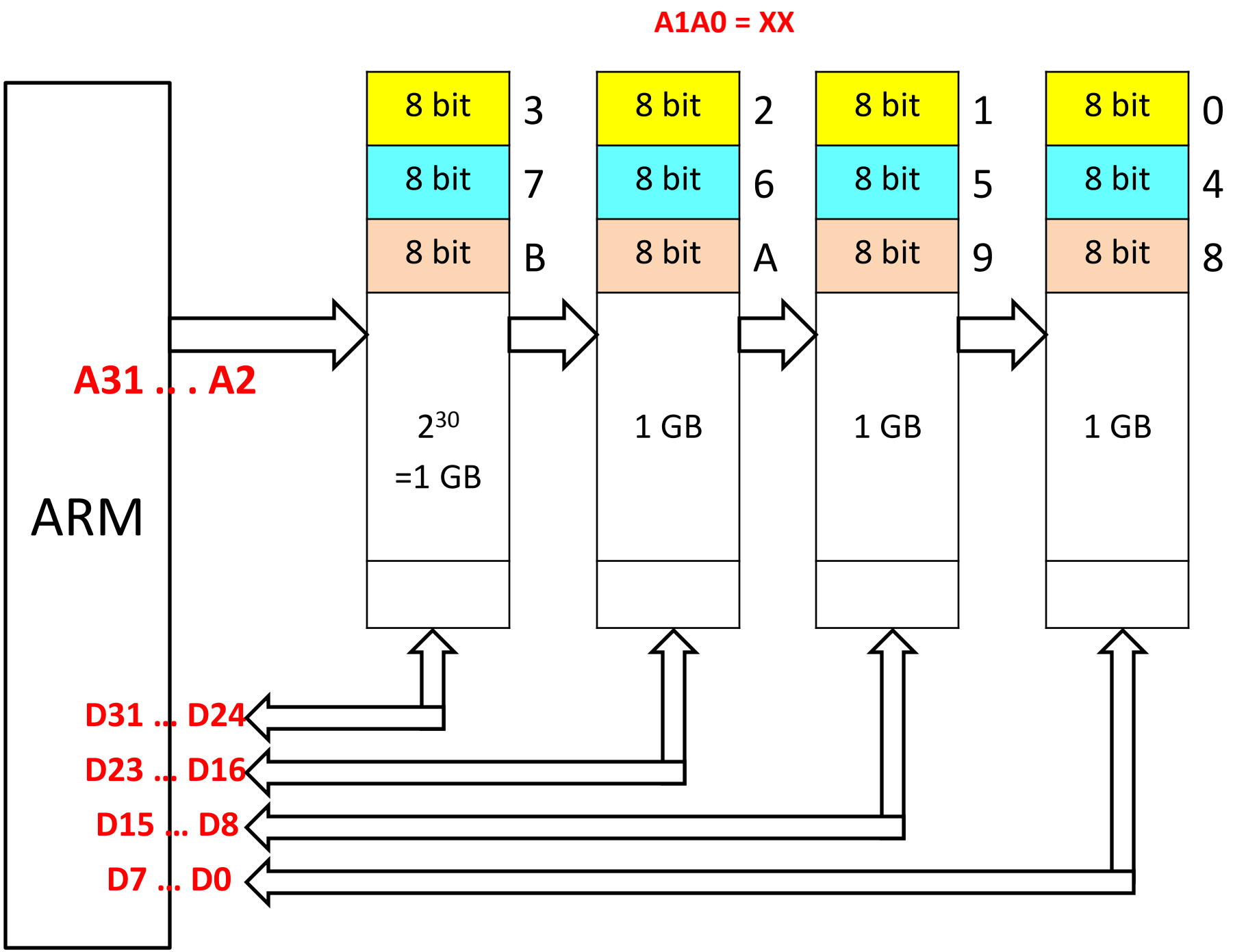


ARM processors features

- It is 32 bit Microcontroller
  - 32 bit ALU
  - 32 bit Registers (R0 to R15)
  - Can ADD, SUB, MUL. DIV, AND, OR, XOR two 32 bit operands
- 32 bit address bus (A31…..A0)
  - Memory capacity 2^{32} = 4GB (4 Giga Bytes)
  - Can select one memory location out of 4GB
- 32 bit data bus (D31 …. D0)
  - Can read / write 32 bit data at once from / to memory
- Memory organized in 4 banks

Memory organization



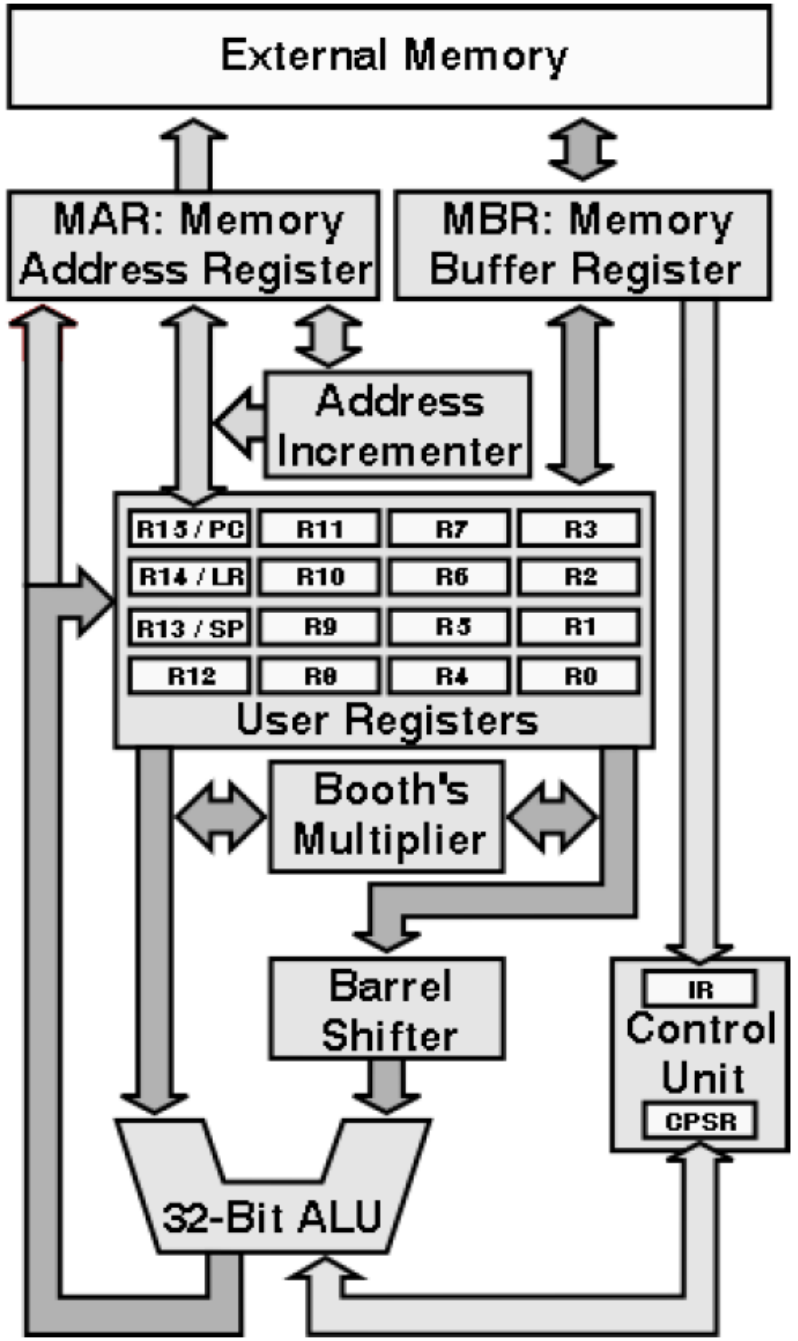

A1A0 = XX
8 bit
3
7
B
8 bit
2
6
A
8 bit
1
5
9
8 bit
0
4
8
A31 .. . A2
ARM
2^{30}
=1 GB
1 GB
1 GB
1 GB
D31 ... D24
D23 ... D16
D15 ... D8
D7 ... D0

ARM processors features

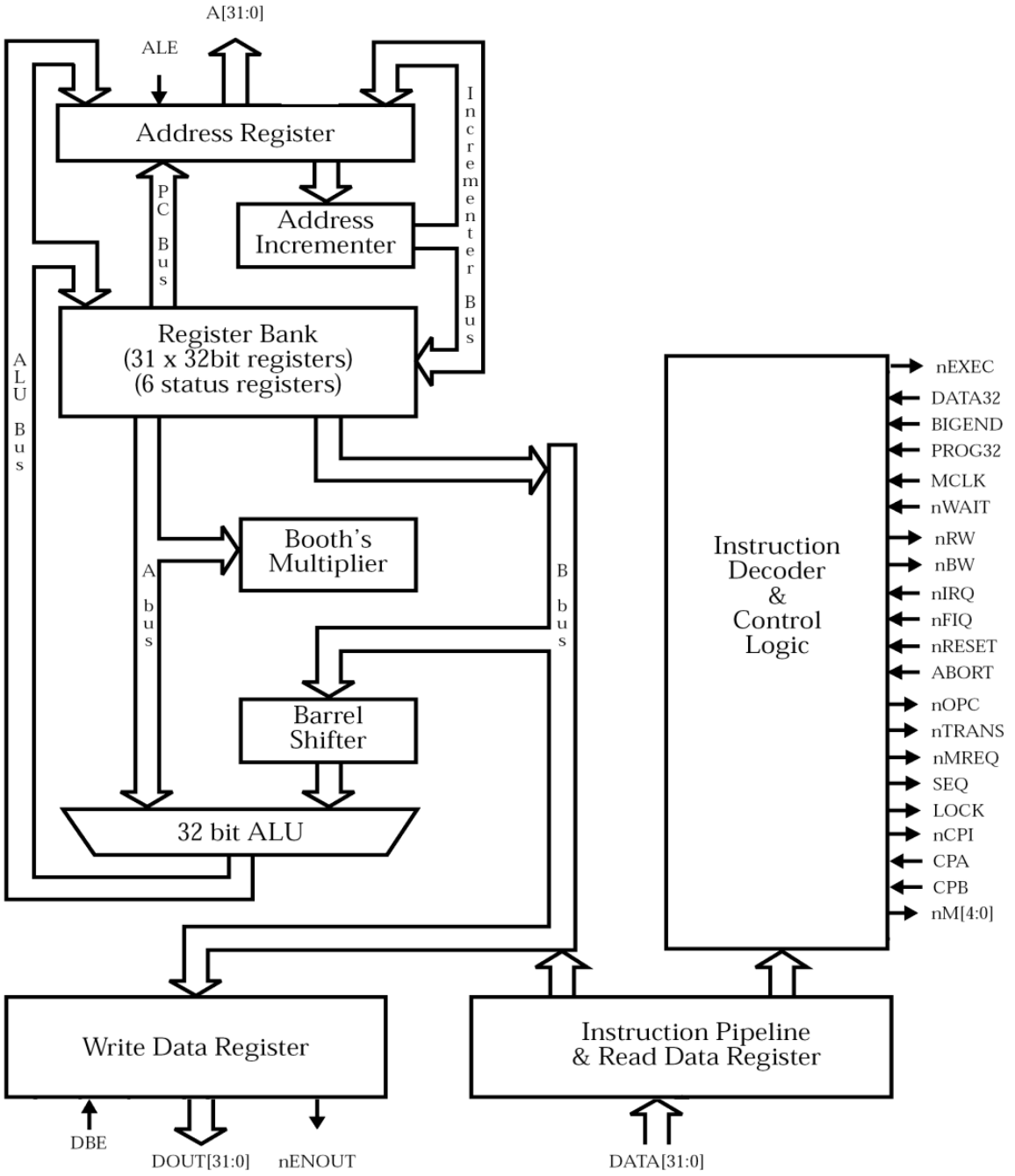
- Load/store architecture
- A large array of uniform registers
- Fixed-length 32-bit instructions
- 3-address instructions (Destination, Operand 1, Operand 2)
- Variable cycle execution for certain instructions: multiple-register load/store (faster/higher code density)
- Inline barrel shifter leading to more complex instructions: improves performance and code density
- Thumb 16-bit instruction set: 30% code density improvement
- Conditional execution: improve performance and code density by reducing branch
- 7 Modes of Operation

ARM Block Diagram

- Arithmetic Logic Unit
- Booth multiplier
- Barrel shifter
- Control unit
- Register file

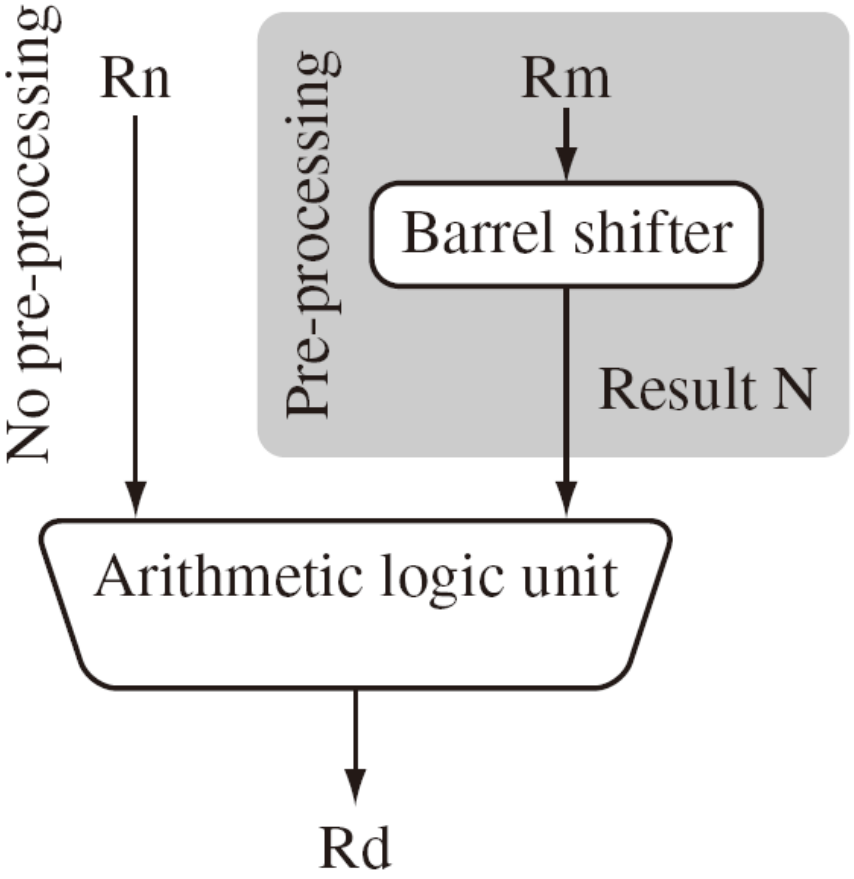


ARM Architecture



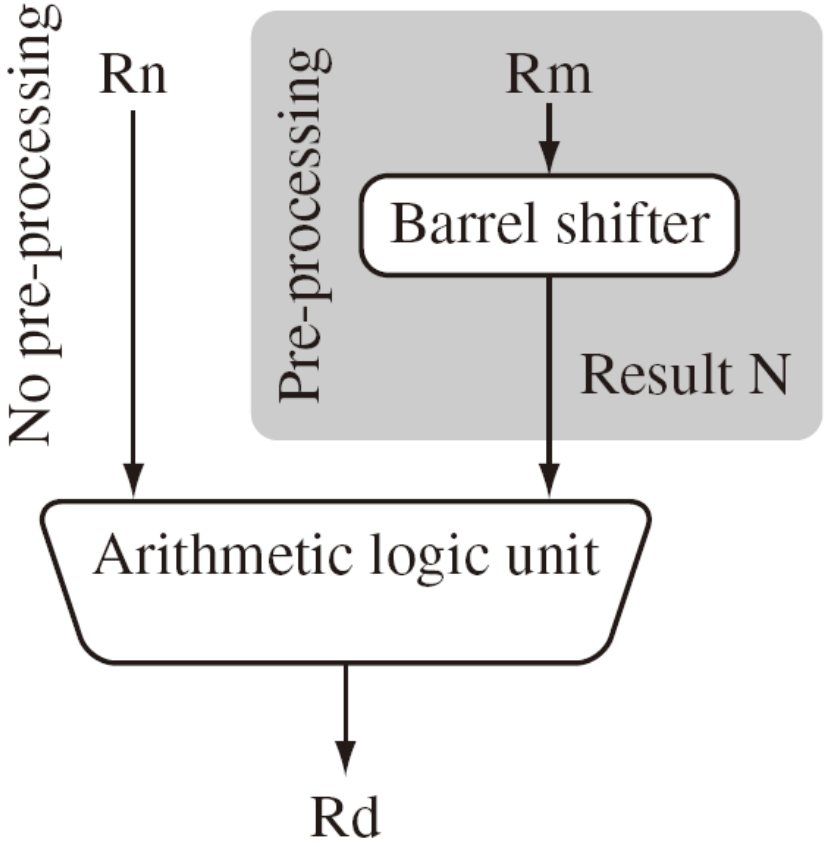
ARM Block Diagram

- Arithmetic Logic Unit (ALU)
- The ALU has two 32-bits inputs.
- The primary comes from the register file, whereas the other comes from the shifter.
- Status registers flags modified by the ALU outputs.
- ADD Rd, Rn, Rm
- ADD R3, R1, R2

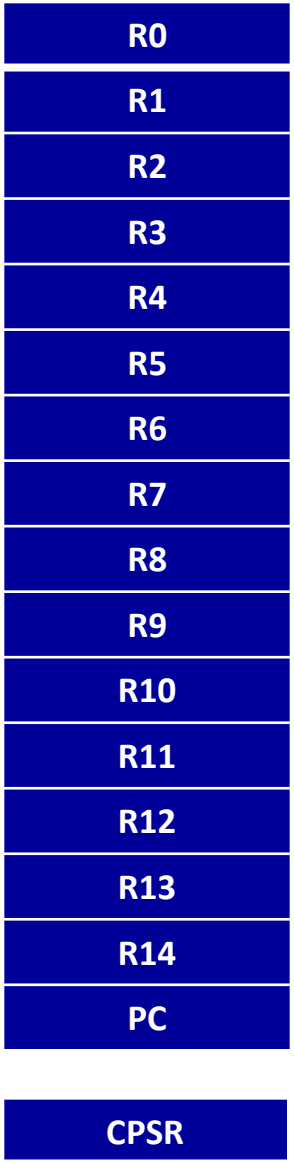


ARM Block Diagram

- Barrel Shifter
- The barrel shifter features a 32-bit input to be shifted. This input is coming from the register file
- logical / arithmetic left or right, shift
- Fast Multiplication/ division
- **R0 = R1 + (R2 * 4)**
- **ADD R0, R1, R2, LSL #2**

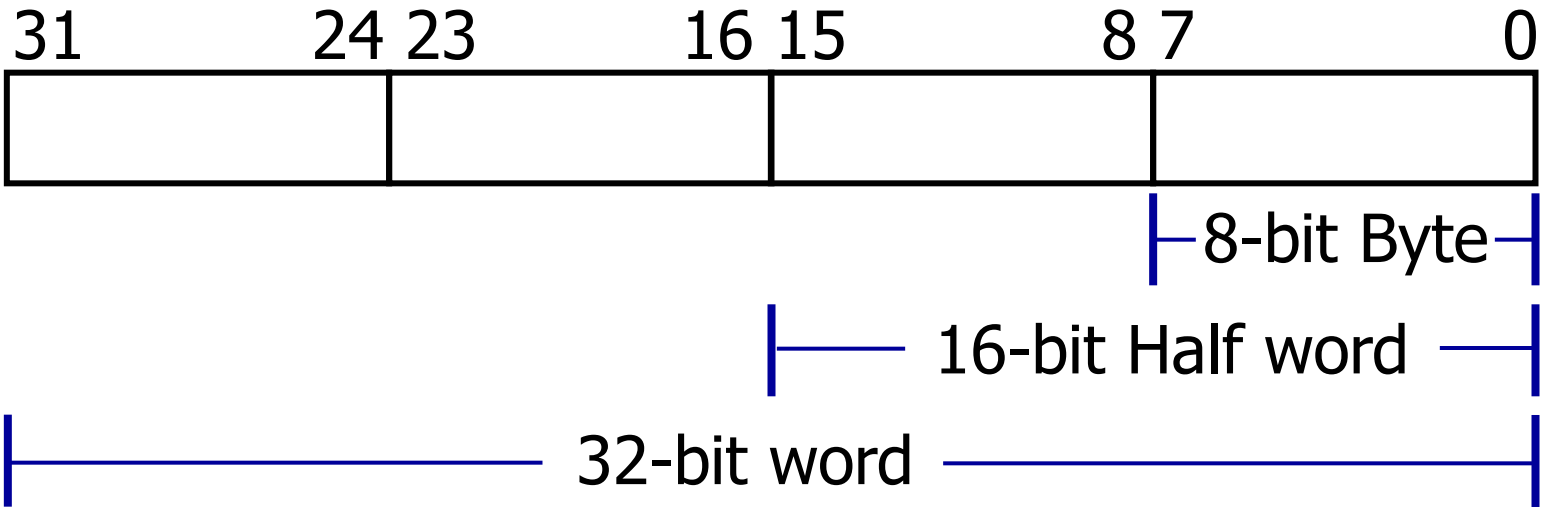


ARM Registers



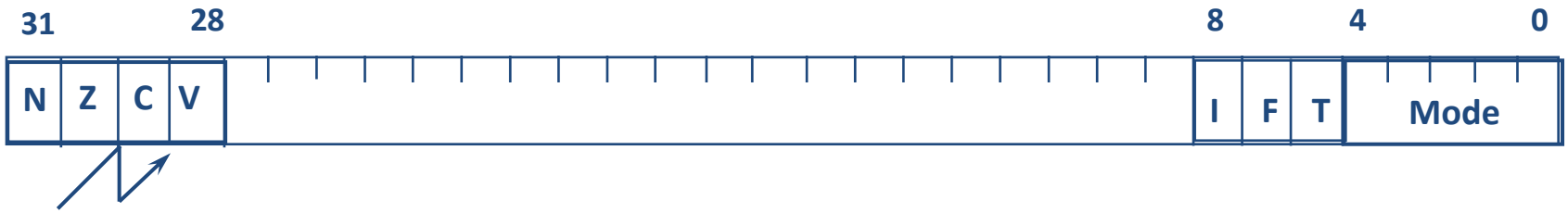
- Only 16 registers are visible to specific mode of operation, each mode can access:
  - A particular set of registers R0 - R12
  - R13 is Stack Pointer (SP)
  - R14 is subroutine Link Register Holds the value of R15 when BL-instruction is executed
  - R15 is Program Counter (PC)
  - Current Program Status Register (CPSR)
- Privileged mode can access particular SPSR
  - Saved Program Status Register (SPSR)

General-purpose registers (R0 – R12)



- 6 data types are supported (signed and un-signed)
- byte, halfword and word
  - 8 bit (byte), 16 bit (half word), 32 bit (word)
- All ARM operations are 32 bit
  - Shorter data types are supported by data transfer operations. load and store operations transfer bytes, halfwords and words to and from memory

Program Status Registers (CPSR & SPSRs)



Copies of the ALU status flags (latched if the instruction has the "S" bit set).

- Condition Code Flags
  - N = Negative result from ALU flag.
  - Z = Zero result from ALU flag.
  - C = ALU operation Carried out
  - V = ALU operation oVerflowed
- Interrupt Disable bits.
  - I = 1, disables the IRQ.
  - F = 1, disables the FIQ.
- T Bit: Processor in ARM (0) or Thumb (1)
- Mode Bits: processor mode

M[4:0]	Mode
10000	User
10001	FIQ
10010	IRQ
10011	SVC
10111	Abort
11011	Undef
11111	System

Status Flags

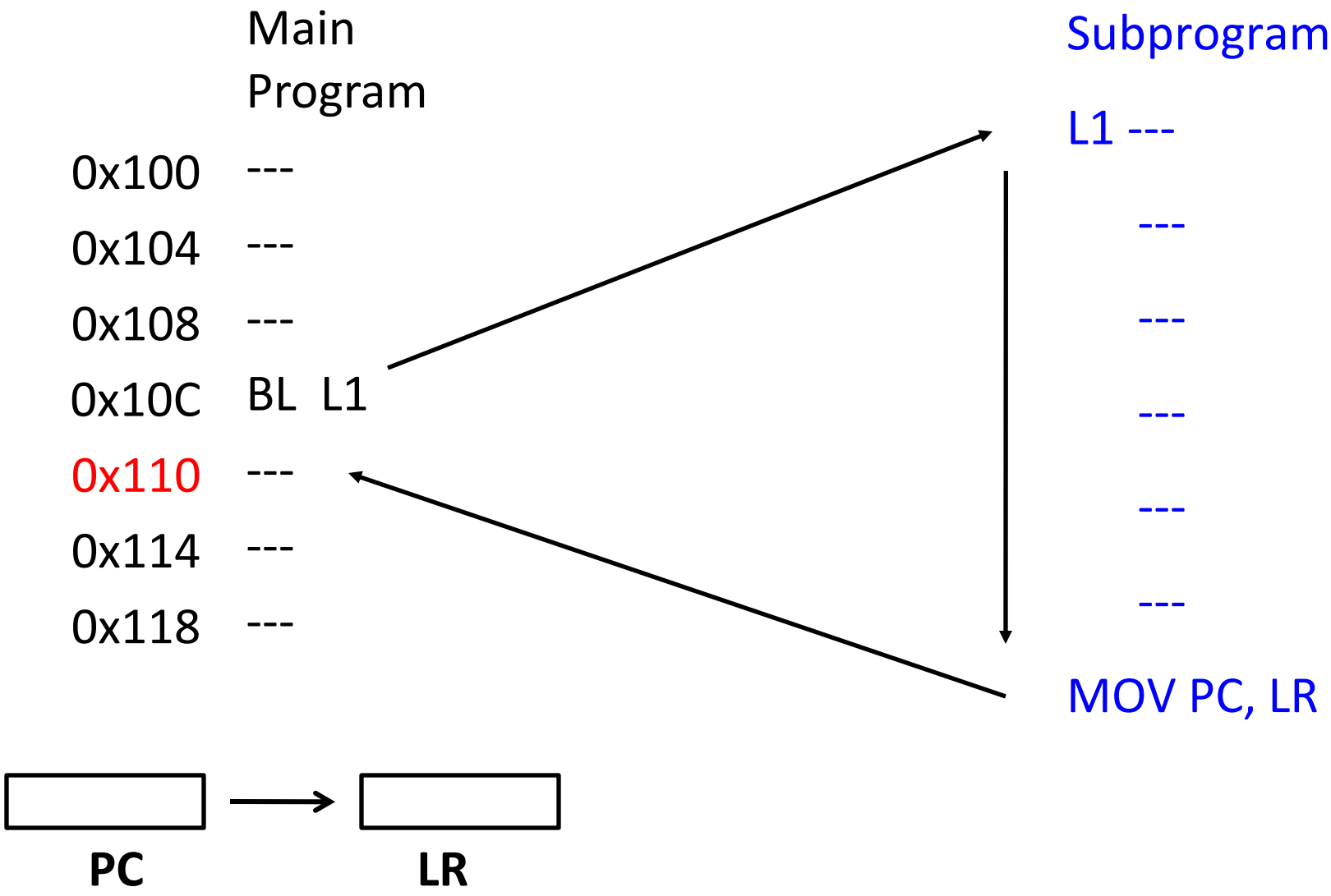
Flag	Logical Instruction	Arithmetic Instruction
Negative (N='1')	No meaning	Bit 31 of the result has been set Indicates a negative number in signed operations
Zero (Z='1')	Result is all zeroes	Result of operation was zero
Carry (C='1')	After Shift operation	Result was greater than 32 bits '1' was left in carry flag
oVerflow (V='1')	No meaning	Result was greater than 31 bits Indicates a possible corruption of the sign bit in signed numbers

The Program Counter (R15)

- When the processor is executing in ARM state:
  - All instructions are 32 bits in length
  - All instructions must be word aligned
  - Therefore the PC value is stored in bits [31:2] with bits [1:0] equal to zero (as instruction cannot be halfword or byte aligned).
- R14 is used as the subroutine link register (LR) and stores the return address when Branch with Link (BL) operations are performed, calculated from the PC.
- Thus to return from a linked branch

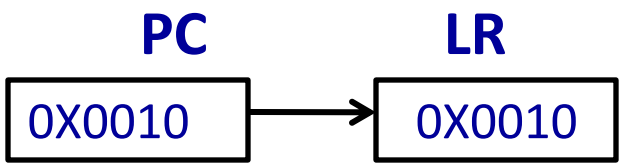
```
MOV r15,r14
MOV pc,lr
```

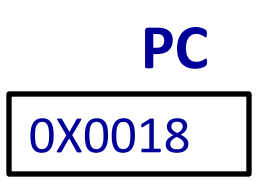
Subroutine call

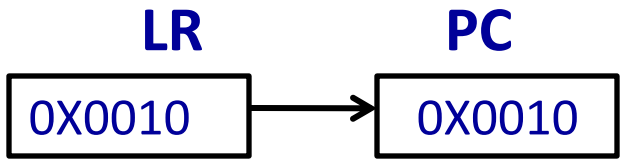


Subroutine call using Branch and Link (BL)

```
PC                AREA  program, code, readonly
                  ENTRY
0X0000            MOV  R1, #0X600
0X0004            MOV  R2, #0X200
0X0008            ADD R0, R1,R2
0X000C            BL  L1
0X0010            MOV  R1, R5
0X0014  L         B    L          ; STOP
0X0018  L1        MOV  R4, #0X300
0X001C            MOV  R6, #0X200
0X0020            SUB  R5, R4, R6
0X0024            MOV  PC, LR
                  END
```





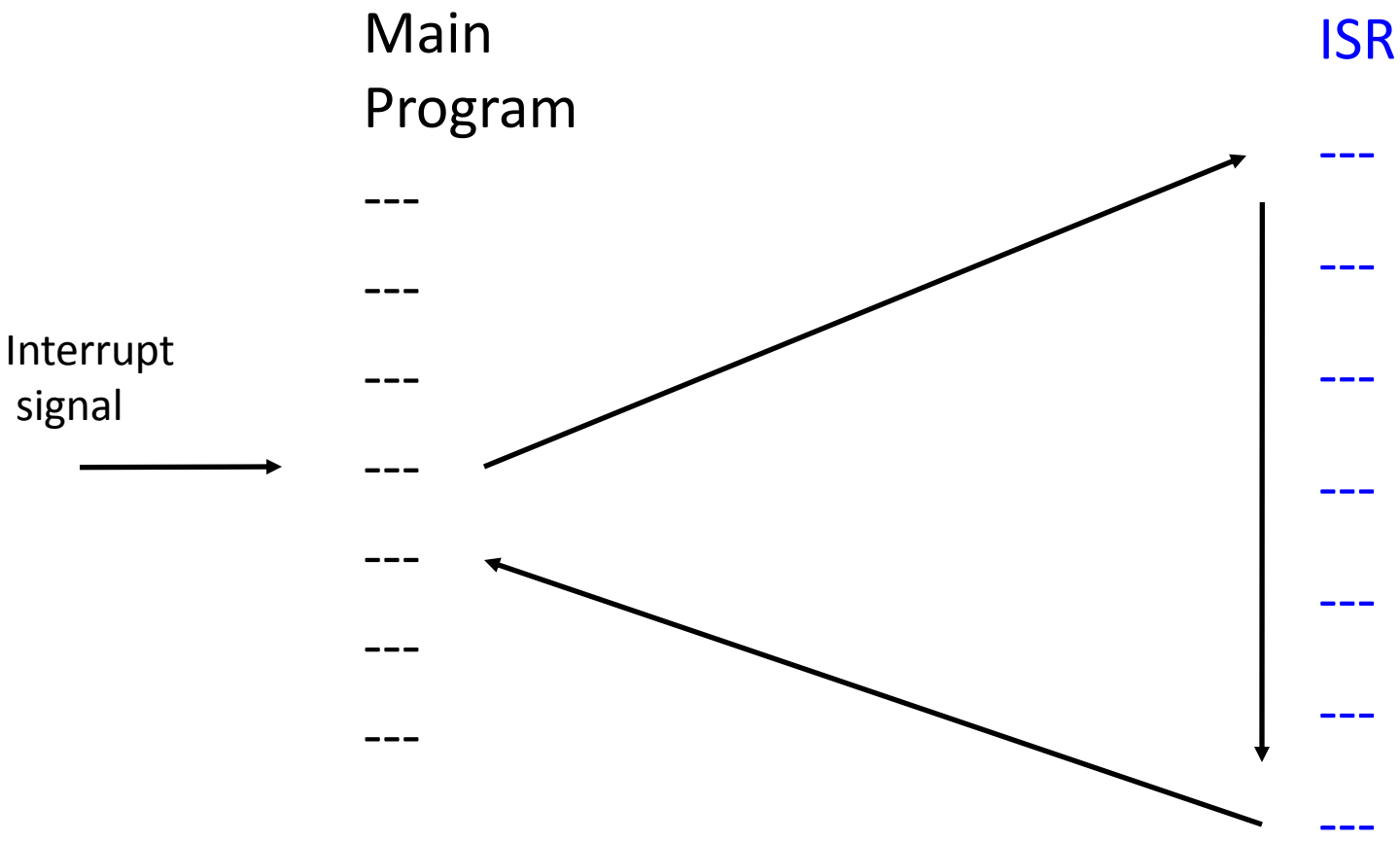


```
      3:          MOV  R1, #0X600
0x00000000  E3A01C06  MOV        R1,#0x00000600
      4:          MOV  R2, #0X200
0x00000004  E3A02C02  MOV        R2,#0x00000200
      5:          ADD R0, R1,R2
0x00000008  E0810002  ADD        R0,R1,R2
      6:          BL  L1
0x0000000C  EB000001  BL         0x00000018
      7:          MOV  R1, R5
0x00000010  E1A01005  MOV        R1,R5
      8: L  B     L              ; STOP
0x00000014  EAFFFFFE  B          0x00000014
      9: L1 MOV  R4, #0X300
0x00000018  E3A04C03  MOV        R4,#0x00000300
     10:          MOV  R6, #0X200
0x0000001C  E3A06C02  MOV        R6,#0x00000200
     11:          SUB  R5, R4, R6
0x00000020  E0445006  SUB        R5,R4,R6
     12:          MOV  PC, LR
0x00000024  E1A0F00E  MOV        PC,R14
```

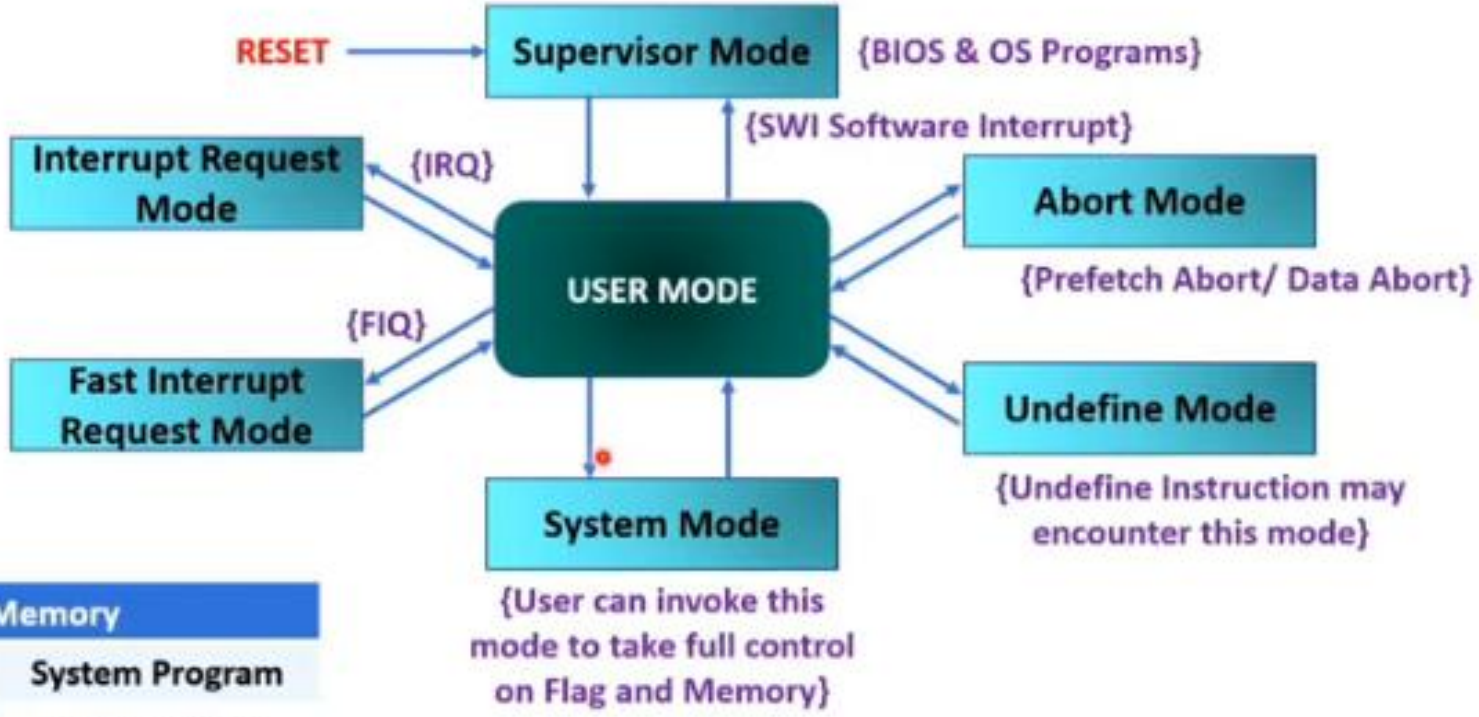
Address	Instruction Code	Instructions	Comments
00000000	E3A01C06	MOV R1, #0X600	
00000004	E3A02C02	MOV R2, #0X200	
00000008	E0810002	ADD R0, R1,R2	
0000000C	EB000001	BL L1	; Go to subroutine at L1
00000010	E1A01005	MOV R1, R5	
00000014	EAFFFFFE	L B L	; STOP
00000018	E3A04C03	L1 MOV R4, #0X300	;Subroutine start here
0000001C	E3A06C02	MOV R6, #0X200	
00000020	E0445006	SUB R5, R4, R6	
00000024	E1A0F00E	MOV PC, LR	Return to main program

Memory Address)	Instruction code (.Hex File)
0x00000000:	06 1C A0 E3
0x00000004:	02 2C A0 E3
0x00000008:	02 00 81 E0
0x0000000C:	01 00 00 EB
0x00000010:	05 10 A0 E1
0x00000014:	FE FF FF EA
0x00000018:	03 4C A0 E3
0x0000001C:	02 6C A0 E3
0x00000020:	06 50 44 E0
0x00000024:	0E F0 A0 E1

Interrupt / Mode switch



ARM7 Operating Modes



Memory	
System	System Program
	System Data
User	User Program
	User Data

Processor Modes

Processor mode			Description
1	User	(usr)	the normal program execution mode
2	FIQ	(fiq)	designed to support a high-speed data transfer or channel process
3	IRQ	(irq)	used for general-purpose interrupt handling
4	Supervisor	(svc)	a protected mode for the operating system
5	Abort	(abt)	used to implement virtual memory and/or memory protection
6	Undefined	(und)	used to support software emulation of hardware coprocessors
7	System	(sys)	used to run privileged operating system tasks (Architecture Version 4 only)

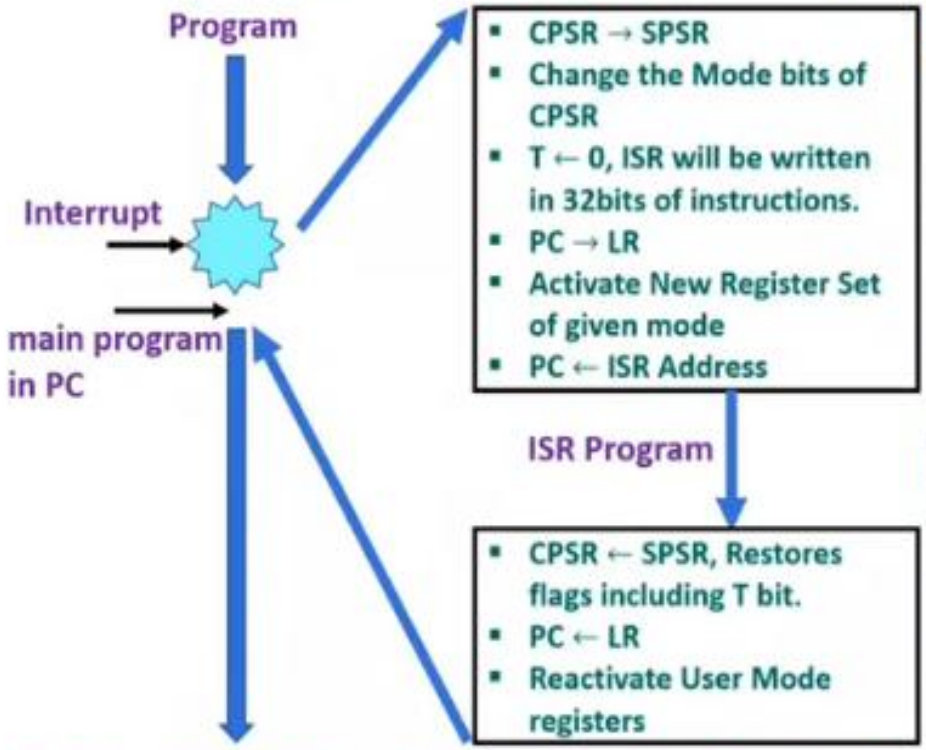
- Mode changes may be made under software control or may be caused by external interrupts or exception processing.
- Most application programs will execute in User mode.
- Other privileged modes will be entered to service interrupts or exceptions or to access protected resources:

ARM7 Operating Modes

Mode Bits					Mode
1	0	0	0	0	User
1	0	0	0	1	FIQ
1	0	0	1	0	IRQ
1	0	0	1	1	Supervisor
1	0	1	1	1	Abort
1	1	0	1	1	Undefined
1	1	1	1	1	System

- User Mode
  - This is normal mode in which all the user programs are executed.
  - It is the only non privileged mode.
  - It has limited access to Memory, IO and Flags.
  - All other modes are entered through interrupt.
- Fast Interrupt Mode
  - This mode is enabled when high priority interrupt comes on nFIQ Pin.
  - This interrupts should be served with minimum delay.
  - By default Nested interrupts are enabled in this Mode.
- Interrupt Mode
  - This mode is enabled when normal interrupt comes on nIRQ Pin.
  - This interrupts will be served with some delay.
  - Nested interrupts are allowed in this Mode.
- Supervisor Mode
  - This mode is enabled when we reset the system.
  - It is used to execute BIOS program.
  - This mode can be invoked by programmer with SWI {Software interrupt}.
- Abort Mode
  - This mode is entered when unsuccessful attempt is made to access memory.
  - Due to protection mechanism some memory locations are not accessible.
- Undefined Mode
  - This mode is entered when undefined instructions attempted.
  - This generally happens when coprocessor instruction encountered but coprocessor is not available.

ARM7 Interrupts / Exceptions



❖ Above steps for Interrupt service is also referred as steps of task switching in ARM7TDMI.

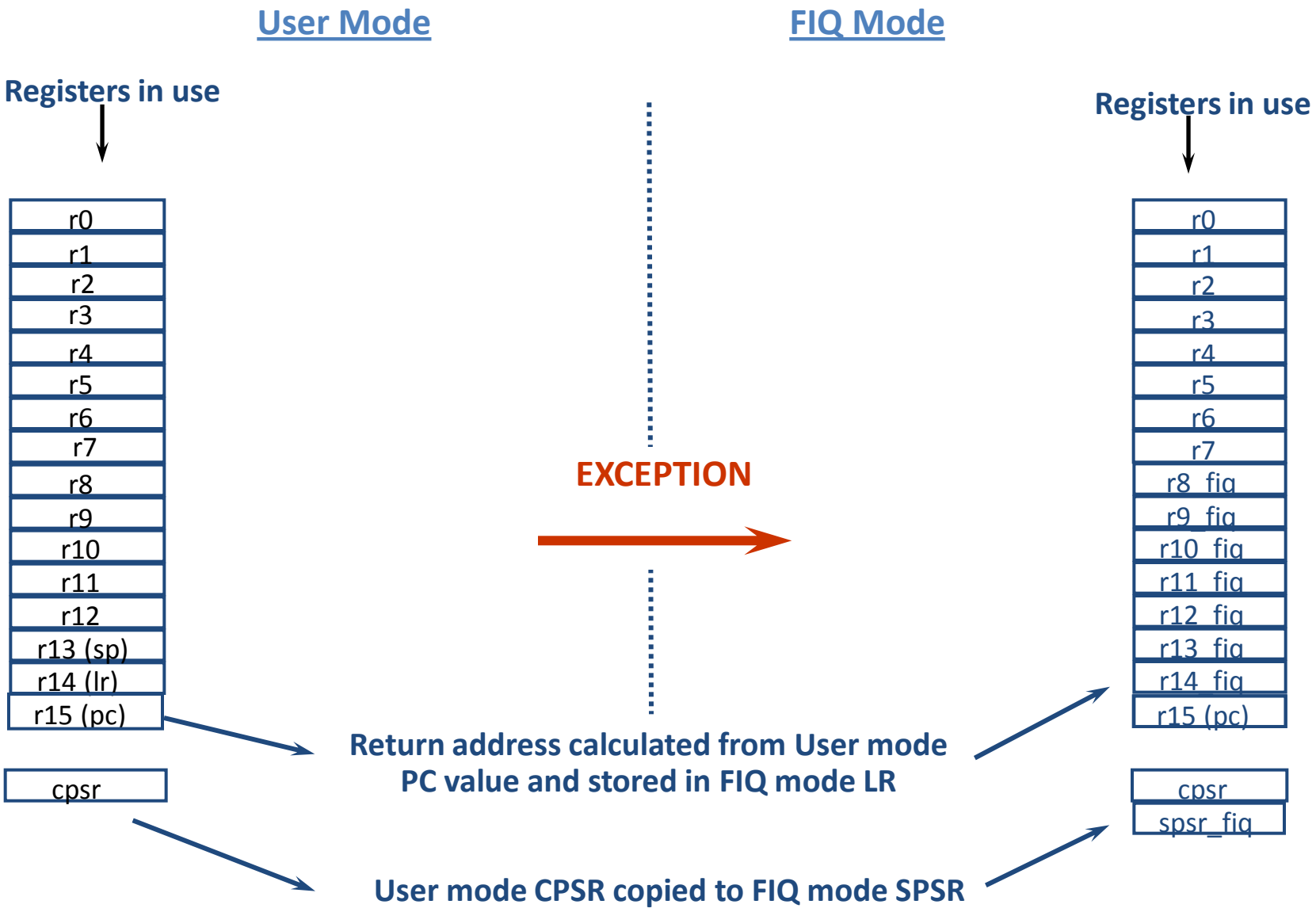
Interrupt	Mode	ISR Normal	ISR High
RESET	Supervisor	0000 0000H	FFFF 0000H
Undefined Instruction	Undefined	0000 0004H	FFFF 0004H
Software Interrupt	Supervisor	0000 0008H	FFFF 0008H
Prefetch Abort	Abort	0000 000CH	FFFF 000CH
Data Abort	Abort	0000 0010H	FFFF 0010H
Reserved by ARM		0000 0014H	FFFF 0014H
IRQ Normal Interrupt	IRQ	0000 0018H	FFFF 0018H
FIQ Fast Interrupt	FIQ	0000 001CH	FFFF 001CH

Interrupt	Priority
RESET	1
Data Abort	2
FIQ Fast Interrupt	3
IRQ Normal Interrupt	4
Prefetch Abort	5
Undefined Instruction	6
Software Interrupt	

Register Organization (37)

User/ sys	SVC	abt	und	irq	fiq
R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8_FIQ
R9	R9	R9	R9	R9	R9_FIQ
R10	R10	R10	R10	R10	R10_FIQ
R11	R11	R11	R11	R11	R11_FIQ
R12	R12	R12	R12	R12	R12_FIQ
R13	R13_SVC	R13_ABORT	R13_UND	R13_IRQ	R13_FIQ
R14	R14_SVC	R14_ABORT	R14_UND	R14_IRQ	R14_FIQ
PC	PC	PC	PC	PC	PC
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_SVC	SPSR_ABORT	SPSR_UND	SPSR_IRQ	SPSR_FIQ

Register Example: User to FIQ Mode



Exception Handling

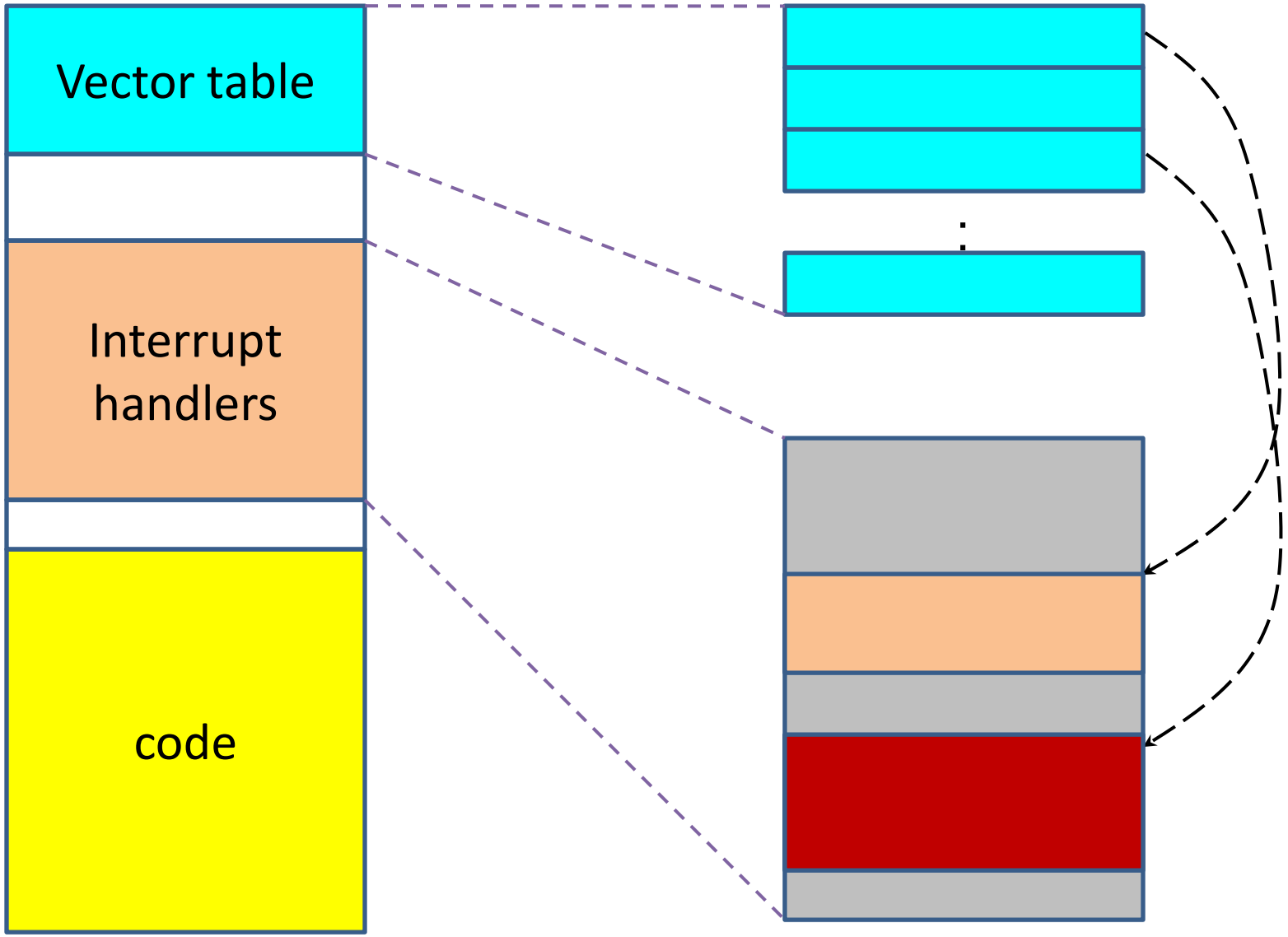
- When an exception occurs,
  - Copies CPSR into SPSR_<mode>
  - Sets appropriate CPSR bits
    - enter ARM state if necessary
    - Mode field bits
    - Interrupt disable flags if appropriate.
  - Maps in appropriate banked registers
  - Stores the “return address” in LR_<mode>
  - Sets PC to vector address

- To return, exception handler needs to:
  - Restore CPSR from SPSR_<mode>
  - Restore PC from LR_<mode>

The Vector Table

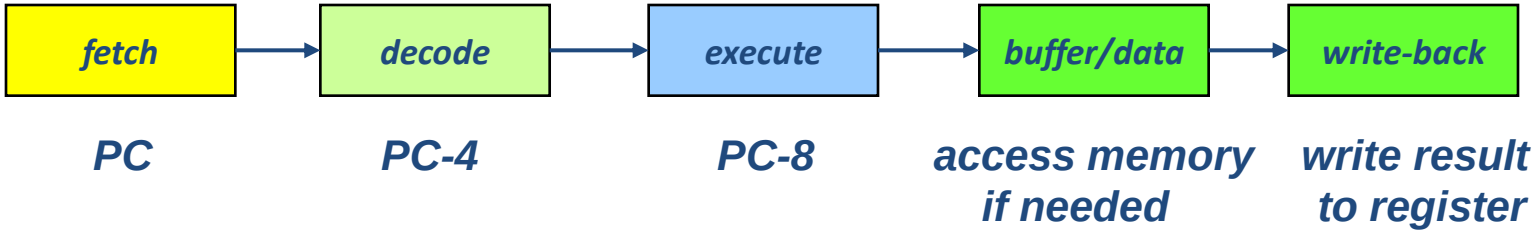
Exception type	Exception mode	Vector address
Reset	Supervisor	0x00000000
Undefined instructions	Undefined	0x00000004
Software Interrupt (SWI)	Supervisor	0x00000008
Prefetch Abort (Instruction fetch memory abort)	Abort	0x0000000c
Data Abort (Data Access memory abort)	Abort	0x00000010
IRQ (Interrupt)	IRQ	0x00000018
FIQ (Fast Interrupt)	FIQ	0x0000001c

Interrupts Service

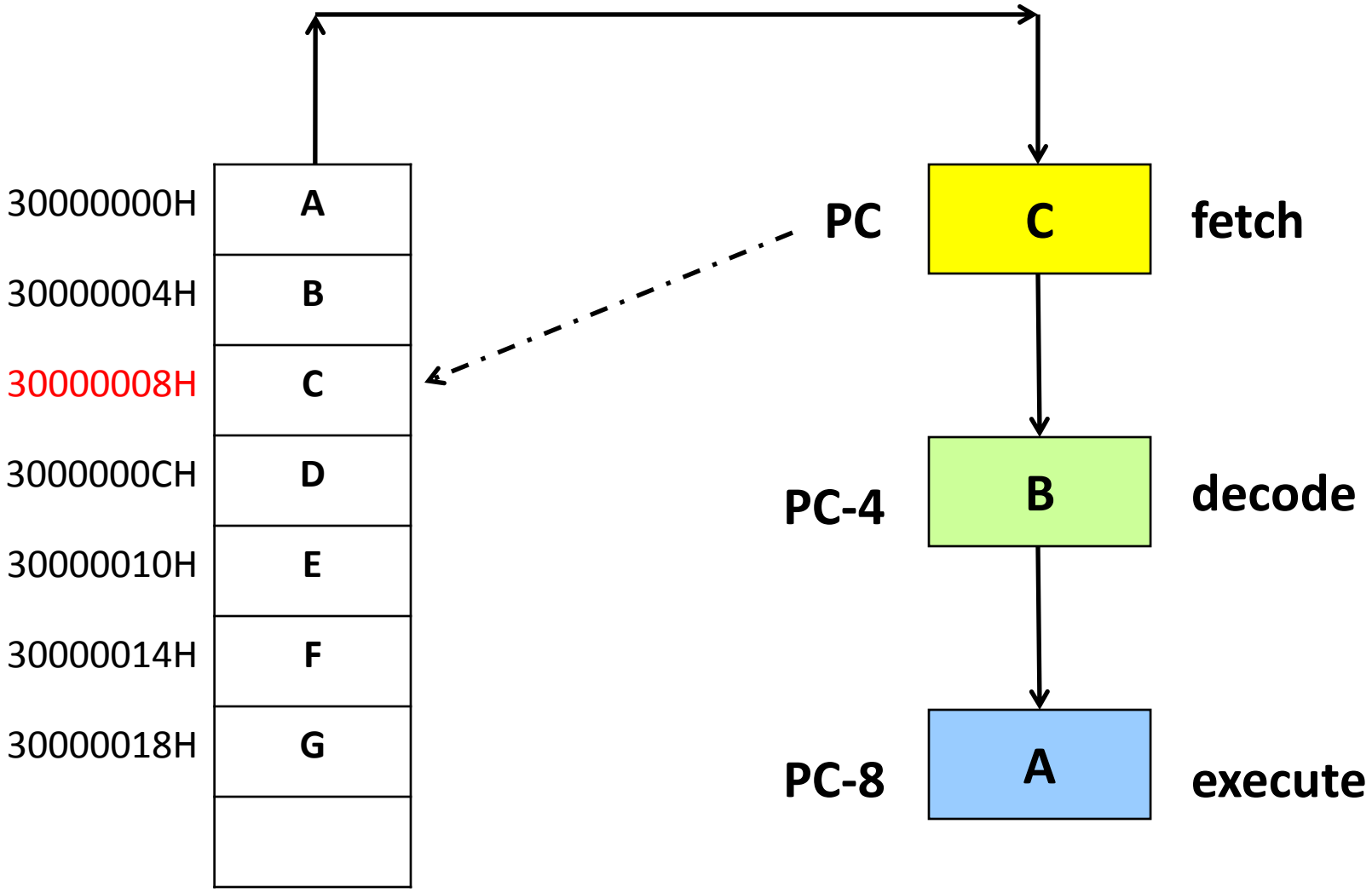


The Instruction Pipeline

- The ARM uses a pipeline in order to increase the speed of the flow of instructions to the processor.
  - Allows several operations to be undertaken simultaneously, rather than serially.
  - PC points to the instruction being fetched.
- 3 stages (ARM7) and 5 stages (ARM9TDMI)



The Instruction Pipeline

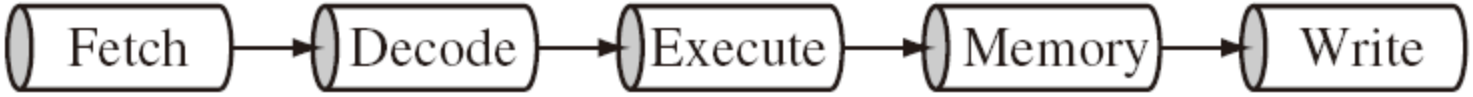


In execution, pc always 8 bytes ahead

Pipeline in ARM

ARM7

ARM9



- Execution of a branch or direct modification of pc causes ARM core to flush its pipeline
- ARM10 starts to use branch prediction
- An instruction in the execution stage will complete even though an interrupt has been raised. Other instructions in the pipeline are abandoned.