Programming the Arm Cortex-M

Dr. Munesh Singh

Indian Institute of Information Technology
Design and Manufacturing,
Kancheepuram
Chennai-600127

January 21, 2019

ARM Mocrocontrollers

- ARM is a family of instruction set architectures for computer processors based on a reduced instruction set computer (RISC) architecture
- It is also known as load-store Architecture:
  - For memory operation microcontroller have to first load desired memory location content, then after CPU operation
- Three varient of ARM microcontroller avaialable in martket:
  - ARM Cortex A: Application Specific
  - ARM Cortex R: Realtime Application
  - ARM Cortex m: General Purpose Application
- ARM holding develops the instruction set and architecture for ARM-based products, but does not manufactured products.

Basic Difference RISC/CISC

RISC

- Compilation side the greater complexity
- Hardware side it is less complex
- RISC Architecture:
  - Instruction
  - Pipelining
  - Registers
  - Load/store architecture

CISC

- Compilation side is less complexity
- Hardware side it is more complex

ARM Design Philosophy

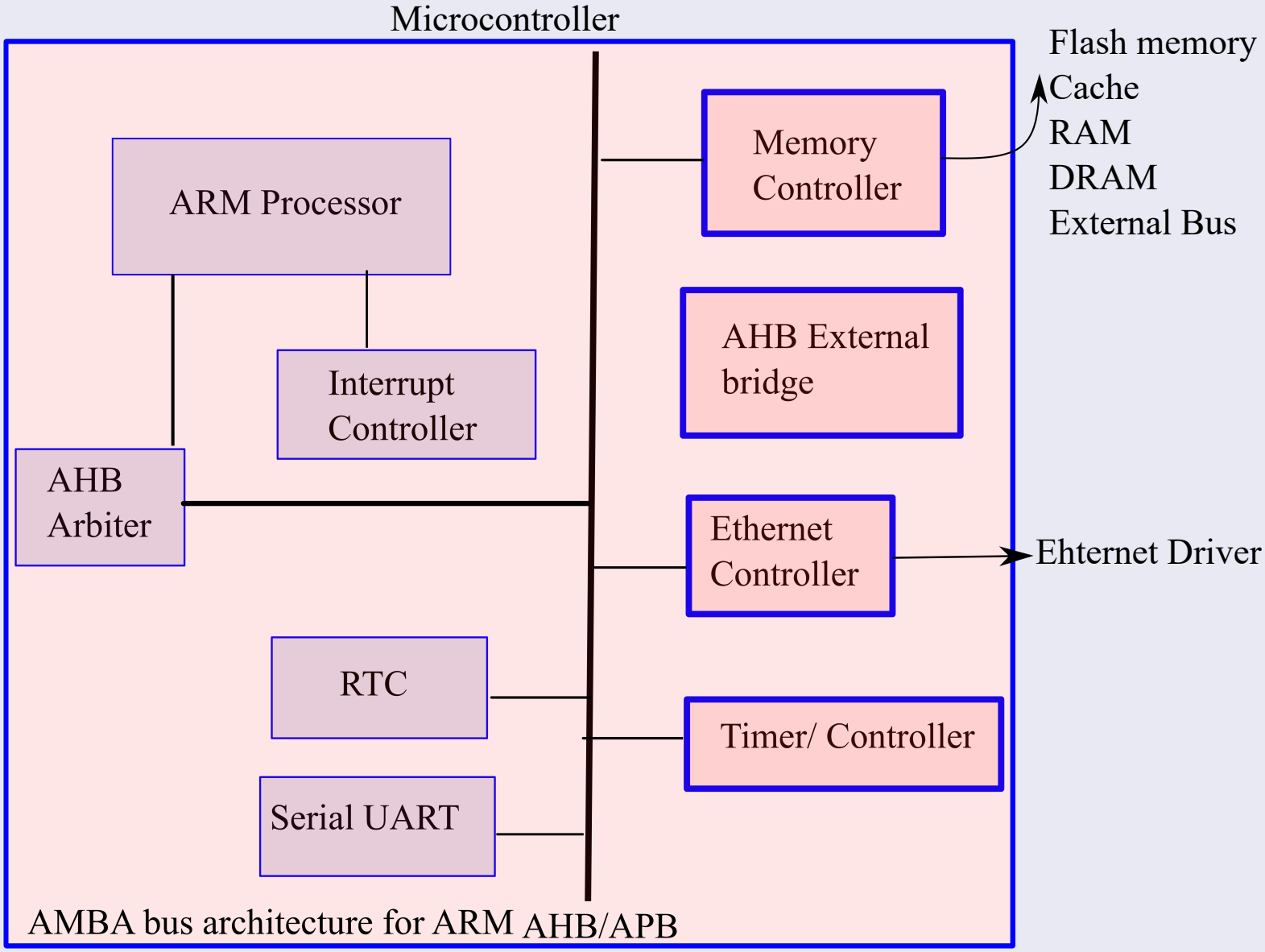
Different from Other Microcontrollers

- High Code Density
- Low Price and Small Form Factor (size of Die)
- Debug technology-JTAG
- Advance RISC machine (not pure RISC)

Instruction Set of ARM

- In RISC only one clock cycle per instruction
- ARM Instruction takes variable clock cycle
  - Load-Store-Multiply
- Inline Barrel Shifter
- Thumb Instruction Set
  - 16 bit thumb instruction set
- Conditional Execution
  - Add if CARRY/OVERFLOW/ZERO
- Enhanced Instructions: (Fast for multiplication)

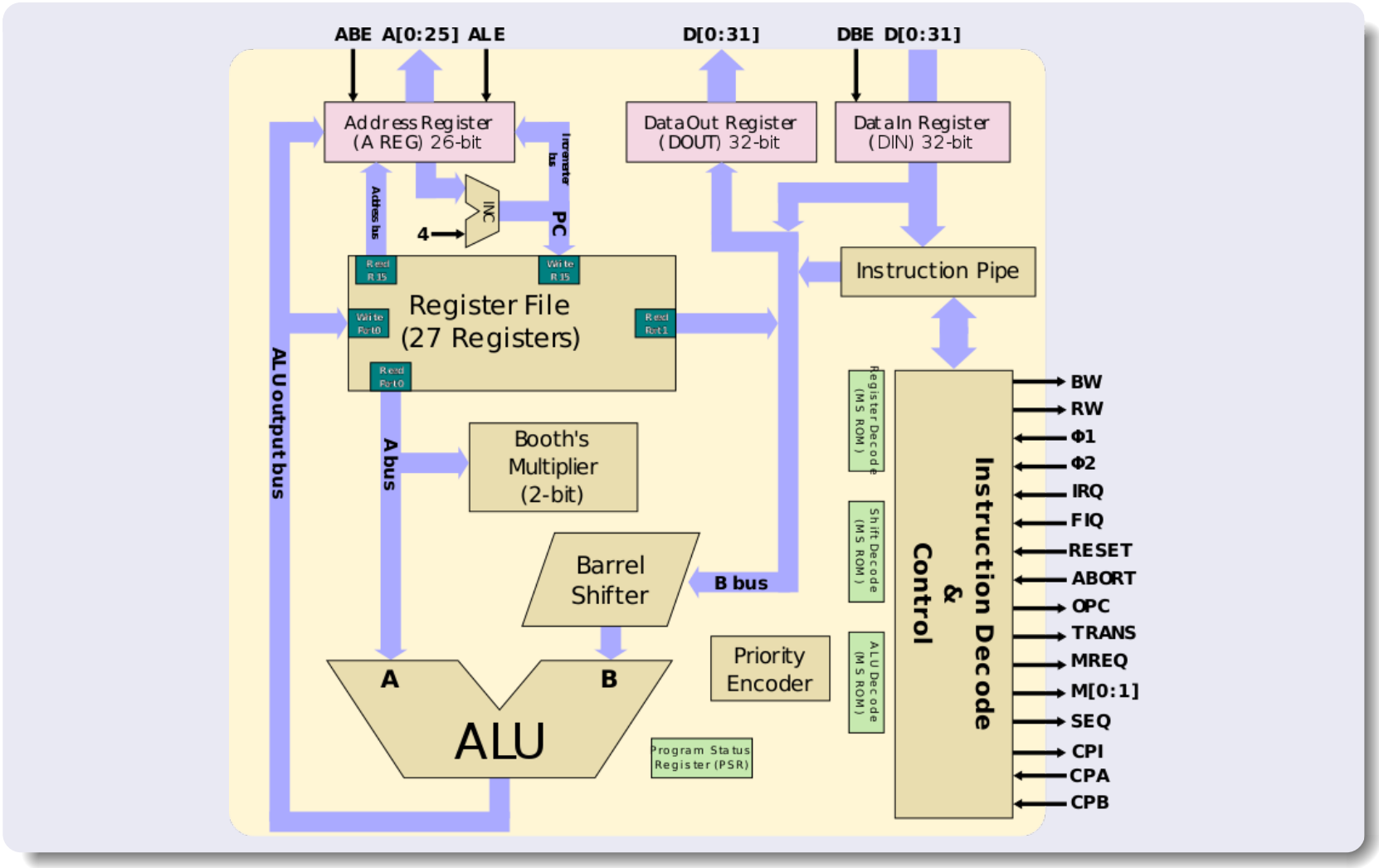
Embedded System Hardware



AMBA Architecture

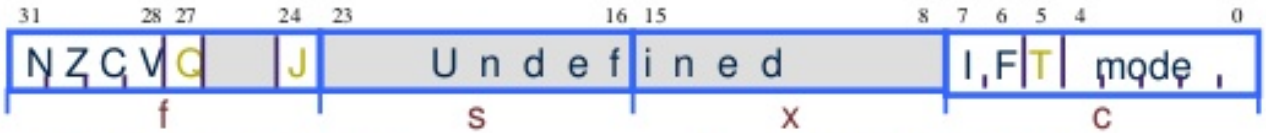
- AMBA bus architecture introduced for ARM Micro-controller
- AMBA stands for Advance Micro-controller BUS Architecture
- It was introduced in 1996, and now it is widely adapted
- AMBA used for micro controller and PCI for Microprocessor
- AMBA is on chip bus architecture, but PCI is external on motherboard
- ARM system Bus (ASB)
- ARM Peripheral bus (APB)
- ARM High Performance BUS (AHB)(high data rate)

Dataflow in ARM



Program Status Register & Modes

ARM Current Program Status Registers



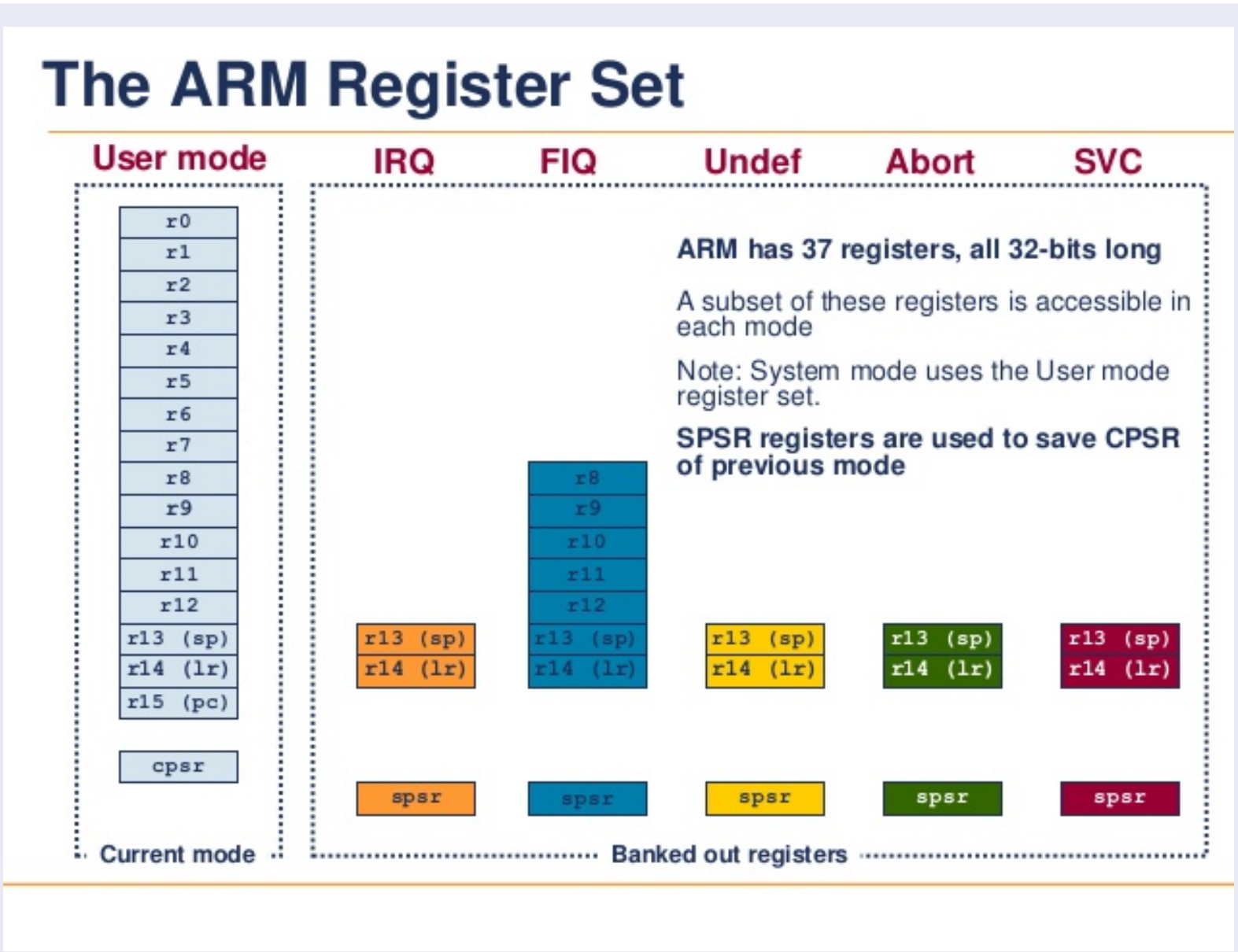
- **Condition code flags**
  - N = Negative result from ALU
  - Z = Zero result from ALU
  - C = ALU operation Carried out
  - V = ALU operation oVerflowed
- **Sticky Overflow flag - Q flag**
  - Architecture 5TE/J only
  - Indicates if saturation has occurred
- **J bit**
  - Architecture 5TEJ only
  - 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

Mode of ARM Processor

7 Types of Modes

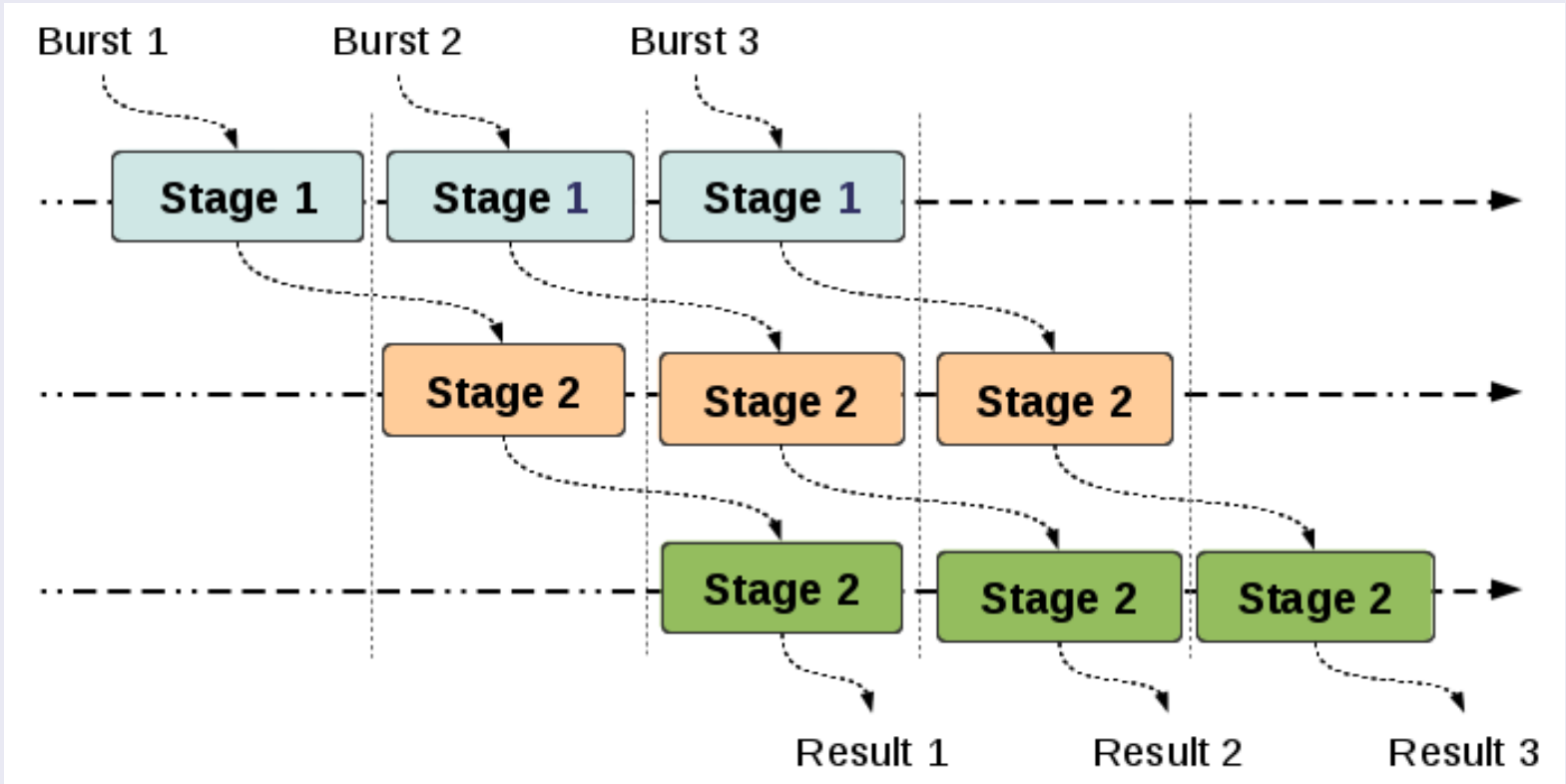
- These modes classified in two categories:
  - Privilege mode[6]
  - Non privileged mode [1]
- In previlege mode:
  - Abort: fault in memory access by processor
  - Fast interrupt request:
  - Interrupt request:
  - Supervised mode:After reset (kernel of processor)
  - System mode: Full read/write to CPSR
  - Undefined mode : Not supported by Architecture of ARM

37 register for all 7 Modes



Pipeline in ARM

ARM7 Three stage pipeline



Data Sizes and Instruction Sets

- The ARM is a 32-bit/64-bit architecture.
- When used in relation to the ARM:
  - Byte mean 8 bits
  - Halfword means 16 bits (two bytes)
  - Word means 32 bits (four bytes)
- Most ARM's implement two instruction sets
  - 32-bit ARM Instruction Set
  - 16-bit Thumb Instruction Set
- Where the processor stores or obtains information
  - Registers
  - Memory
  - Input/Output devices
- How the processor manipulates data
  - Assembly language instructions
  - Processor hardware actions

Introduction of instruction Set

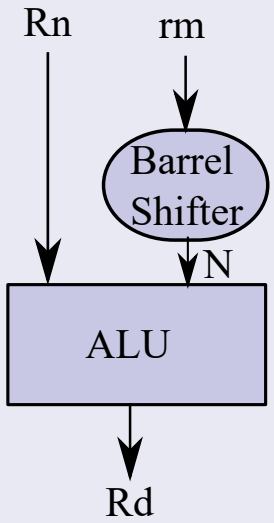
MOV instruction

- Syntax: $<$ instruction $>$ $\{Cond\}$ $\{S\}$ Rd, N
- Example:
- r5=5, r7=8
- mov r7, r5;
- r5=5, r7=5

Barrel Shifter

Barrel Shift Preprocessing command

LSL (logical shift left)
LSR(logical shift right)
ASR(Arithmetic shift right)
ROR(Rotate right)
RRX(Rotate right extended)



r5=5
r7=8
mov r7, r5, LSL #2;

r5=00000101
r5=00010100
r7=r5<<2

r5=5
r7=20

Instruction Set cont...

S suffix in Instruction

- S suffix update the CPSR register
- CPSR=$nzcvqift_{user}$
- r0=0x00000000
- r1=0x80000004
- movs r0,r1,LSL #1
- CPSR=$nzCvqift_{user}$
- Capital C in CPSR register shows carry occurred
- r0=0x00000008
- r1=0x80000004
- Update of status register useful for conditional instruction

Instruction Set cont...

Conditional Instructions

- Condition field two letter mnemonics
  - AL-Always
  - EQ-Equal
  - GT-greater than
  - LT-Less than
  - BAL-Branch Always
  - BEQ-Branch if equal
- An exmple with ADD and EQ instruction
- ALU operation was performed with suffix s
- ADDEQ r0,r1,r2;
- Only comparison and data processing instructions update status reg with added suffix S in instruction
- Comparison and data processing instruction can only update the program status register

Instruction Set cont...

Conditional Instructions in C

- if a==b
- then jump to test

Conditional code in Assembly

- r0=a
- r1=b
- cmp r0,r1;
- BEQ test
- test is label where compiler jump

Thumb Mode vs ARM Mode Instrucion Set

Thumb instrution is 16 bits

- Thumb instruction not allowed ADDEQ

ARM instrution is 32 bits

- ARM instruction allowed ADDEQ

GCD program in C

- while(a!=b) {
- if(a>b)
- a=a-b;
- else
- b=b-a;
- }

Comparison of Thumb vs ARM

Thumb 16 bit

1. gcd
2. cmp r1,r2
3. BEQ Complete
4. BLT less than
5. SUB r1,r1,r2;
6. B gcd
7. less than
8. SUB r2,r2,r1;
9. B gcd
10. Complete r1,r2
11. code size = 7 instruction*2byte=14 byte

ARM 32 bit

1. gcd
2. cmp r1,r2
3. SUBGT r1,r2,r2
4. SUBLT r2,r2,r1
5. BNE gcd
6. gcd r1,r2
7. code size = 4 instruction*4byte=16 byte

Airthmetic Instruction

- Syntax: $< instruction > \{cond\}\{S\}$ Rd, Rn, Rm
- **ADC**(Add Carry) Rd,Rn,N;− > Rd=Rn+N+Carry
- **ADD**(Addition) Rd, Rn, N;− > Rd=Rn+N
- **RSB**(Reverse Subtract) RSB Rd Rn N;− > Rd=N-Rn
- **SUB** (Sub) Rd,Rn,N;− > Rd=Rn-N
- **SBC** (Sub with carry) Rd,Rn,N;− > Rd=Rn-N-!(carry flag)
- **RSC**(Reverse Sub with carry) Rd, Rn, N;− > Rd=N-Rn-!(carry flag)

Airthmetic Instruction Example

SUB

1. r0=0x00000000
2. r1=0x00000077
3. RSB r0,r1,#0;
4. r0=0-r1
5. r0=0xffffff89

SUBS

1. r1=0x00000001
2. SUBS r1,r1,#1
3. r1=r1-1 =0x00000000
4. CPSR=nzcv
5. Post exe:0110

RSB

1. r0=0x00000000
2. r1=0x00000002
3. r2=0x00000001
4. SUB r0,r1,r2
5. r0=r1-r2
6. r0=0x00000001

Logical Instruction

- AND Rd,Rn,N $->$ Rd=Rn & N
- ORR Rd,Rn,N $->$ Rd= Rn |N
- EOR Rd,Rn,N $->$ Rd=Rn $\wedge$ N
- BIC Rd,Rn,N $->$ Rd=Rn&($\sim$N)
- The logical instruction updates the SPSR flags only if "S" suffix is present

ORR

- r0=0x00000000
- r1=0x02040608
- r2=0x10305070
- ORR r0,r1,r2
- r0=r1|r2
- r0=0x12345678

BIC (Bit clear)

- r1=0b1111
- r2=0b0101
- BIC r0,r1,r2
- r0=r1&($\sim$r2)
- r0=0b1010

Comparison Instruction

- Syntax $< instruction > \{cond\}$ Rn,N
- It has no suffix s even it automatically update flags
- **CMN** (compare negate) cmn Rn,N;$- >$ Rn+N
- **CMP** (compare) cmp Rn,N; $- >$ Rn-N
- CMN and CMP do arithmetic operation for comparison
- **TEQ** (test equivalent) TEQ Rn,N;$- >$ Rn$\wedge$N
- **TST** (test bit for 32 bit values) TST Rn,N;$- >$ Rn & N
- TEQ and TST do logical operation for comparison

Example

- r0=4
- r1=4
- cmp r0,r1; r0-r1=Z status register 1

Multiply Instruction

- Syntax: MLA $\{cond\}$ $\{S\}$Rd,Rm,Rs,Rn $->$ Rd=(Rm*Rs)+Rn
- MUL $\{cond\}$ $\{S\}$ Rd, Rm, Rs $->$ Rd=Rm*Rs
- if result gets more than 32 bit, then result will be stored in register pair
- MUL instruction is frequently used in DSP for filtration

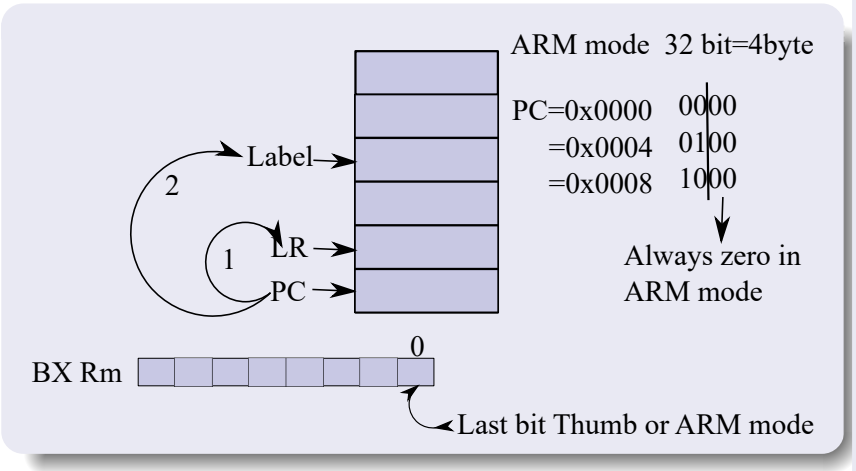
Example UMULL

- r0=0x00000000
- r1=0x00000000
- r2=0xf0000002
- r3=0x00000002
- UMULL r0,r1,r2,r3;
- (r1,r0)=r2*r3

Signed and Unsigned Multiply

- SMLAL(Signed multiply accumulate long)
- SMULL(Signed multiply long)
- UMLAL (Unsigned muliply accumulate long)
- UMULL (Unsigned multiply long)

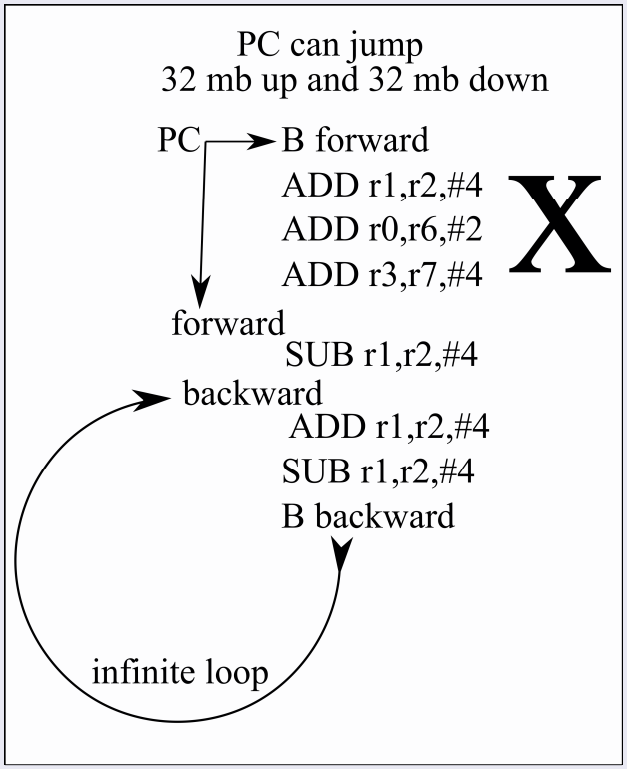
Branch Instruction

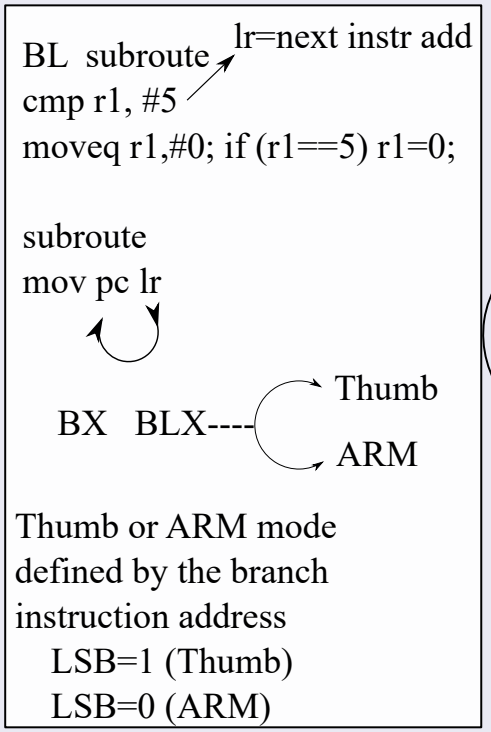


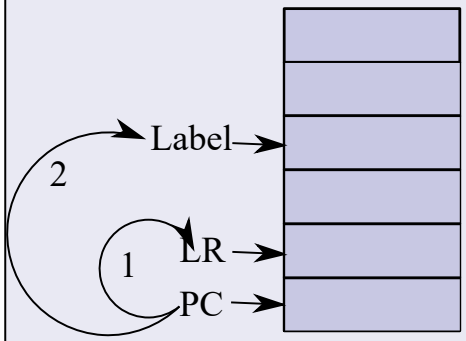
- Syntax: B {*Cond*} label $->$ PC=Label
- BL{*Cond*} label $->$ LR=Address of next instruction
- BX {*Cond*} Rm $->$ Branch and Instruction mode change (Thumb, ARM)
- BLX {*Cond*} label | Rm

Branch Instruction Cont..

Branch Instruction Example







Load and Store Instruction

Single Register Transfer

- Syntax LDR {*Cond*} SB|H|SH Rd, addressing mode
- SB− > Single byte
- H − > Half word
- SH − > Signed Half Word
- STR {*Cond*} H Rd, addressing

Addressing Mode

1. Preindex with writeback
2. Preindex
3. Post index

Load and Store Instruction

Types of Load and Store Instruction

- LDR
- STR
- LDRB
- STRB
- LDRH
- STRH
- LDRSB
- LDRSH

Addressing Mode

1. Preindex with writeback
2. Preindex
3. Post index