CSE-3103: Microprocessor and Microcontroller

Dept. of Computer Science and Engineering
University of Dhaka

Prof. Sazzad M.S. Imran, PhD
Dept. of Electrical and Electronic Engineering
sazzadmsi.webnode.com

RISC Processor

RISC vs CISC →

CISC processor has large and diverse set of instructions,
some instructions can perform multiple operations or access memory directly.

RISC processor has smaller and simpler set of instructions,
each instruction can perform only one operation,
instructions work only on registers rather than memory.

RISC processor can execute more instructions per clock cycle,
it may need more instructions to perform same task as CISC processor.

RISC Features →

- large number of general-purpose registers →
  - can store data and operands,
- uniform and fixed instruction format →
  - easy decoding and pipelining,
- simple and regular instruction set →
  - encoded with fewer bits and executed faster.

RISC Processor

RISC Features →

- instructions can only operate on registers,
- separate load and store instructions to access memory.
- hardware-based branch prediction mechanism →
  - guesses outcome of conditional branches,
  - reduces risk of pipeline stalls.

RISC Examples →

ARM with 32-bit or 64-bit instruction set and extensions,
multiple coprocessors for multimedia, security, signal processing.

MIPS is primarily for networking, gaming, embedded applications.
32-bit or 64-bit instruction set,
multiple instruction sets and coprocessors.

PowerPC was for personal computers, servers, consoles.
32-bit or 64-bit instruction set and extensions,
multiple coprocessors for floating-point, vector, parallel processing.

ARM = Advanced RISC Machine
MIPS = Microprocessor without Interlocked Pipeline Stages

ARM7 Architecture

Advanced RISC Machine series 7 →

- power efficient processors,
- applications →
  - wireless telecommunications,
  - data communication (protocol converter),
  - portable instruments,
  - portable computers and smart cards.

Features →

- 32-bit microprocessor,
- capable of running 16-bit instruction set →
  - achieve greater code density,
  - enhanced power saving.
- register oriented load-and-store architecture.
- can operate in a number of operating modes.
- simple design and low gate count.

ARM7 Architecture

Features →

built-in JTAG debug port (Joint Test Action Group interface) +
on-chip “embedded ICE” (In-Circuit Emulator).
these 2 allows →

- programs to be downloaded directly to chip,
- full debugging while system is running (in-system),
- setting breakpoints, watchpoints.
- tracing program execution step-by-step.

code and data regions are accessed via single data bus.
register-to-register data processing instructions are single-cycle.
data transfer instructions are multi-cycle.

Single-cycle →

ADD, SUB, AND, ORR, MOV, CMP

Multi-cycle →

LDR (load register from memory)
STR (store register to memory)

<u>ARM7 Architecture</u>

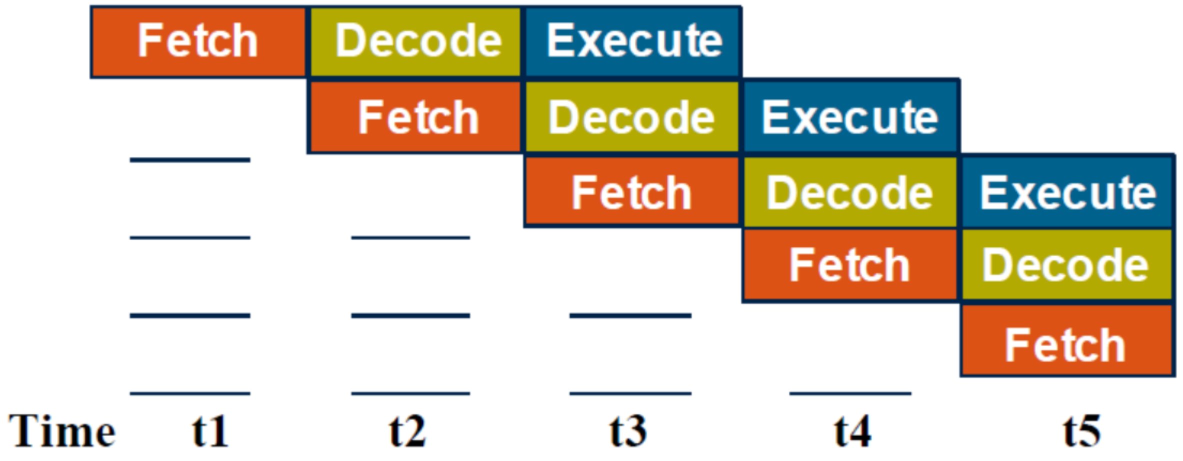
3 stage pipeline →

FETCH, DECODE, EXECUTE.

hardware of each stage is independent →

up to 3 instructions can be processed simultaneously.

concept of 3 stage pipelined execution →



ARM7 Architecture

Programming Model (Register set) →

15 user registers +

R15 = Program Counter (PC).

user program loads data from memory into CPU registers,

processes this data,

stores result back into memory.

R13 is used as stack pointer,

stack handling →

- does not have PUSH and POP instructions.
- set of instructions is used for loading and storing of multiple registers.

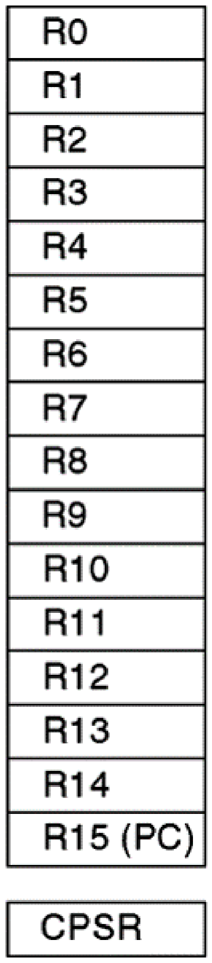
R14 = link register.

CALL is made to procedure →

- return address is automatically placed into R14.
- return is implemented by moving contents of R14 to R15 (PC).

nested subroutine →

- R14 is placed onto stack.



ARM7 Architecture

Programming Model (Register set) →

16 CPU registers + current program status register (CPSR).

CPSR →

upper 4 bits →

set of condition code flags,

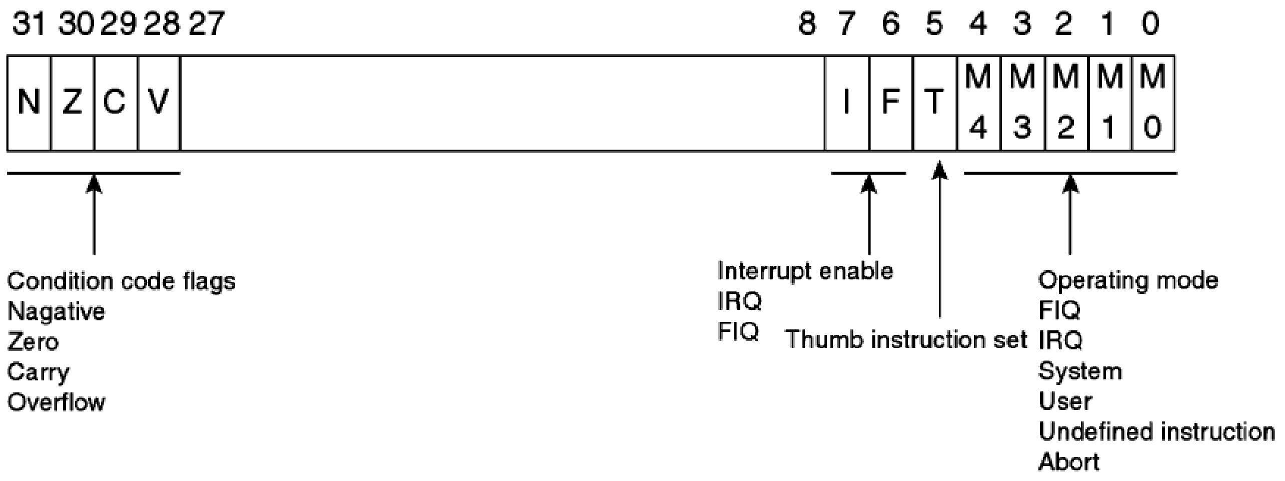
records result of previous instruction.

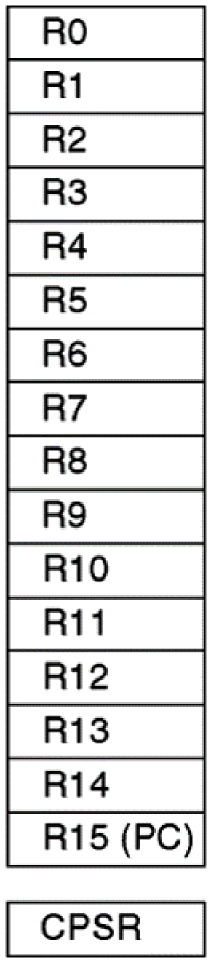
number of user-configurable bits →

change processor mode,

enter Thumb processing,

enable/disable interrupts.





ARM7 Architecture

Programming Model (Register set) →

CPSR →

N (negative)	1 = result of operation is -ve	0 = result of operation is +ve.
Z (zero)	1 = result of operation is zero	0 = result of operation is not zero
C (carry)	1 = result of operation generated carry	0 = result of operation did not produce carry
V (overflow)	1 = result of operation generated overflow	0 = result of operation did not generate overflow

Control bits →

I (interrupt bit): 1 = disable interrupt,
processor does not accept any software interrupt.

F: disable and enable fast interrupt request mode (FIQ).

M4, M3, M2, M1, M0: mode status bits,
= 10000 for user mode.

T (state bit): 1 = processor executing Thumb instruction.
0 = processor executing ARM instruction.

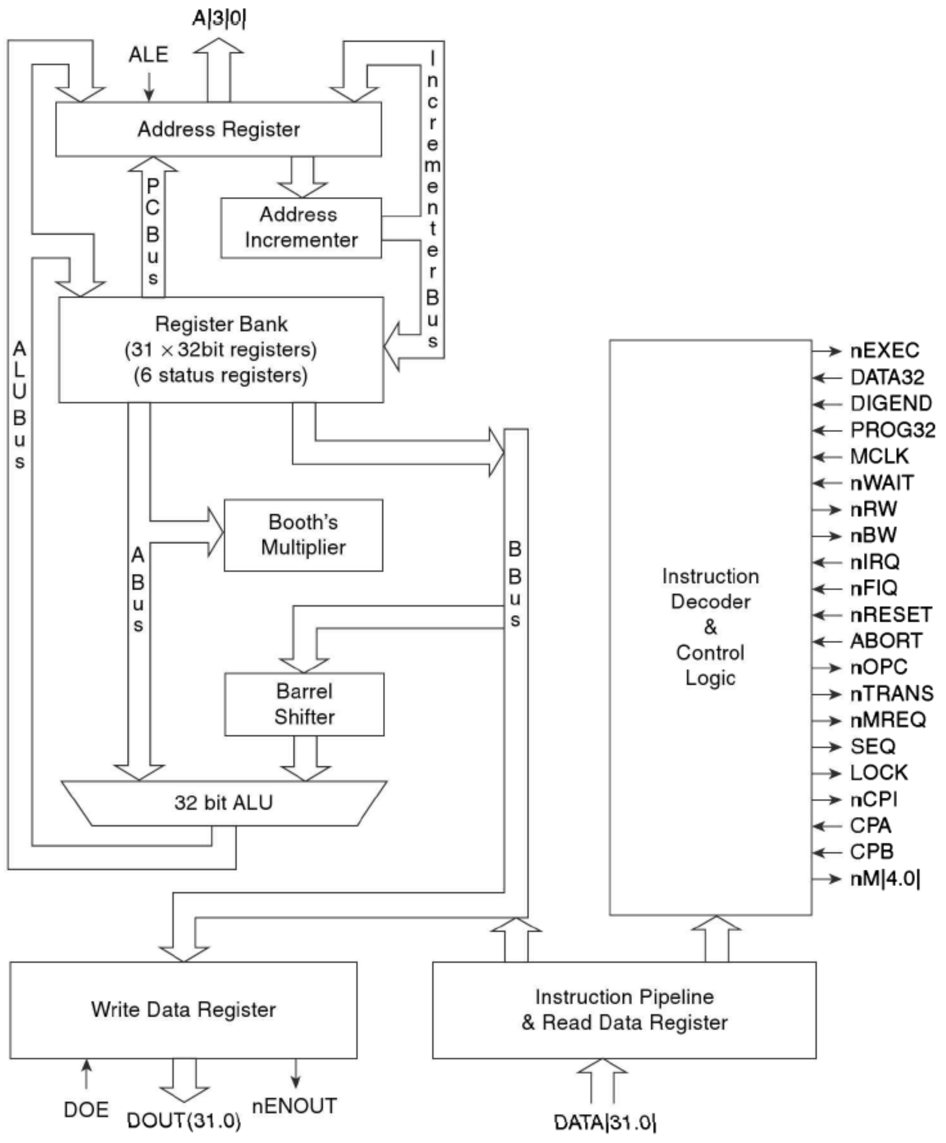
ARM7 Architecture

Load and store architecture →
Instruction decoder and logic control →
fetch and decode instructions,
execution of instructions →
generate control signals to other
parts of processor and external circuits.

Address register →
hold 32-bit address for sending to
address bus.

Address incrementer →
increment addresses by 4 and
place it in address register.

Register bank →
contains thirty one 32-bit registers +
6 status registers.



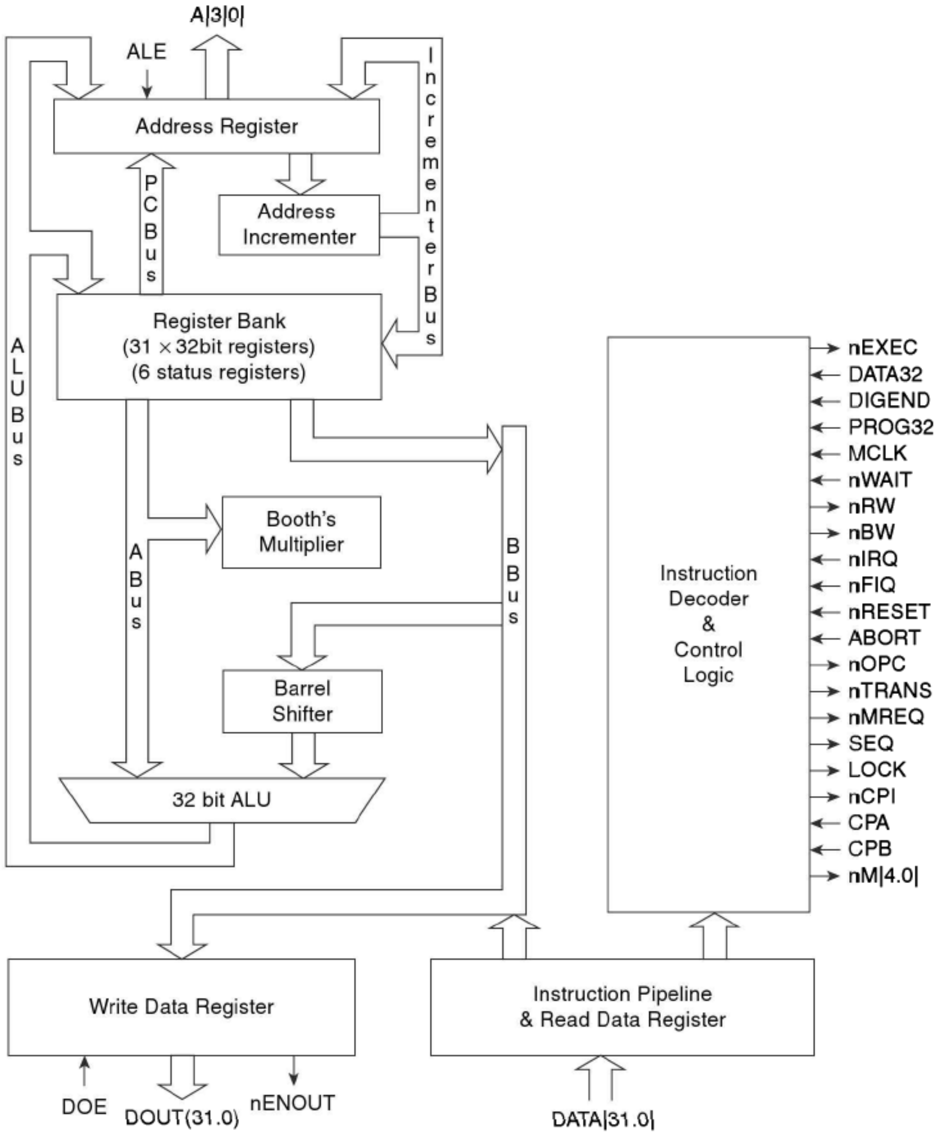
ARM7 Architecture

Load and store architecture →
Booths multiplier + Barrel shifter →
used for fast multiplication and
shift operation by large number of bits.

ALU →
32-bit ALU is used for arithmetic and
logic operations.

Write/Read data register →
processor stores data in Write Data
Register (DR) for write operation.
processor reads from memory or I/O →
places data in Read Data Register.

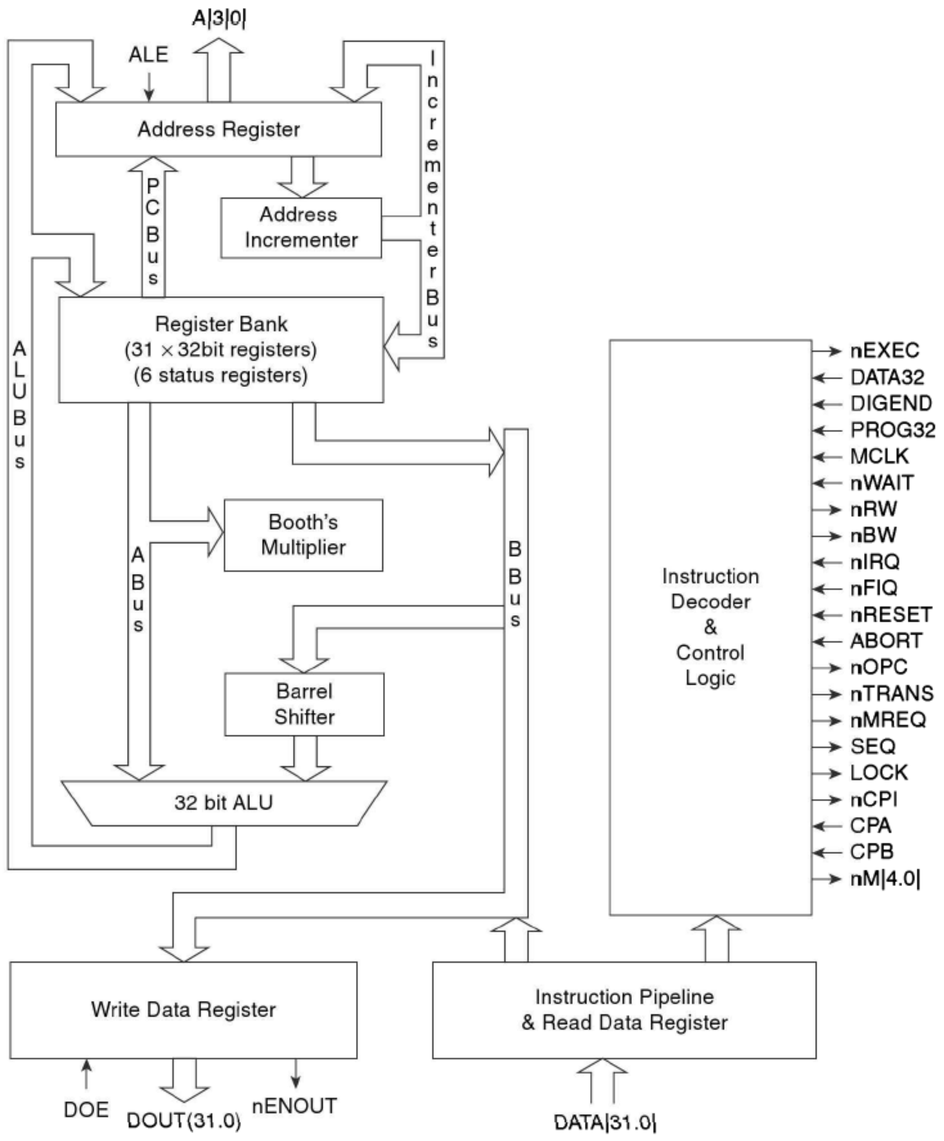
Large number of busses →
enhanced parallelism in architecture.



ARM7 Architecture

Load and store architecture →
Enhanced speed of execution →
non-multiplexed address and data bus,
separate read and write data bus.

Separate operand busses A and B for
ALU operands →
eliminate use of temporary registers.
Separate ALU output bus +
A and B operand busses achieve overlap of →
writing of results of current operation
into register bank +
fetching operands of next operation
from register bank.



ARM7 Architecture

Exception and interrupt modes →

6 different operating modes.

1) User →

used to run application code.

once in user mode →

CPSR cannot be written to.

modes can only be changed when exception is generated.

2) FIQ (fast interrupt request) →

supports high speed interrupt handling.

used for single critical interrupt source in system.

3) IRQ (interrupt request) →

supports all other interrupt sources in system.

4) Supervisor →

protected mode for running system level code to
access hardware or run OS calls.

ARM7 Architecture

Exception and interrupt modes →

5) Abort →

abort exception is generated if
instruction or data is fetched from invalid memory region.

6) Undefined instruction →

UI exception will be generated if
fetched opcode is not ARM instruction.

Register set →

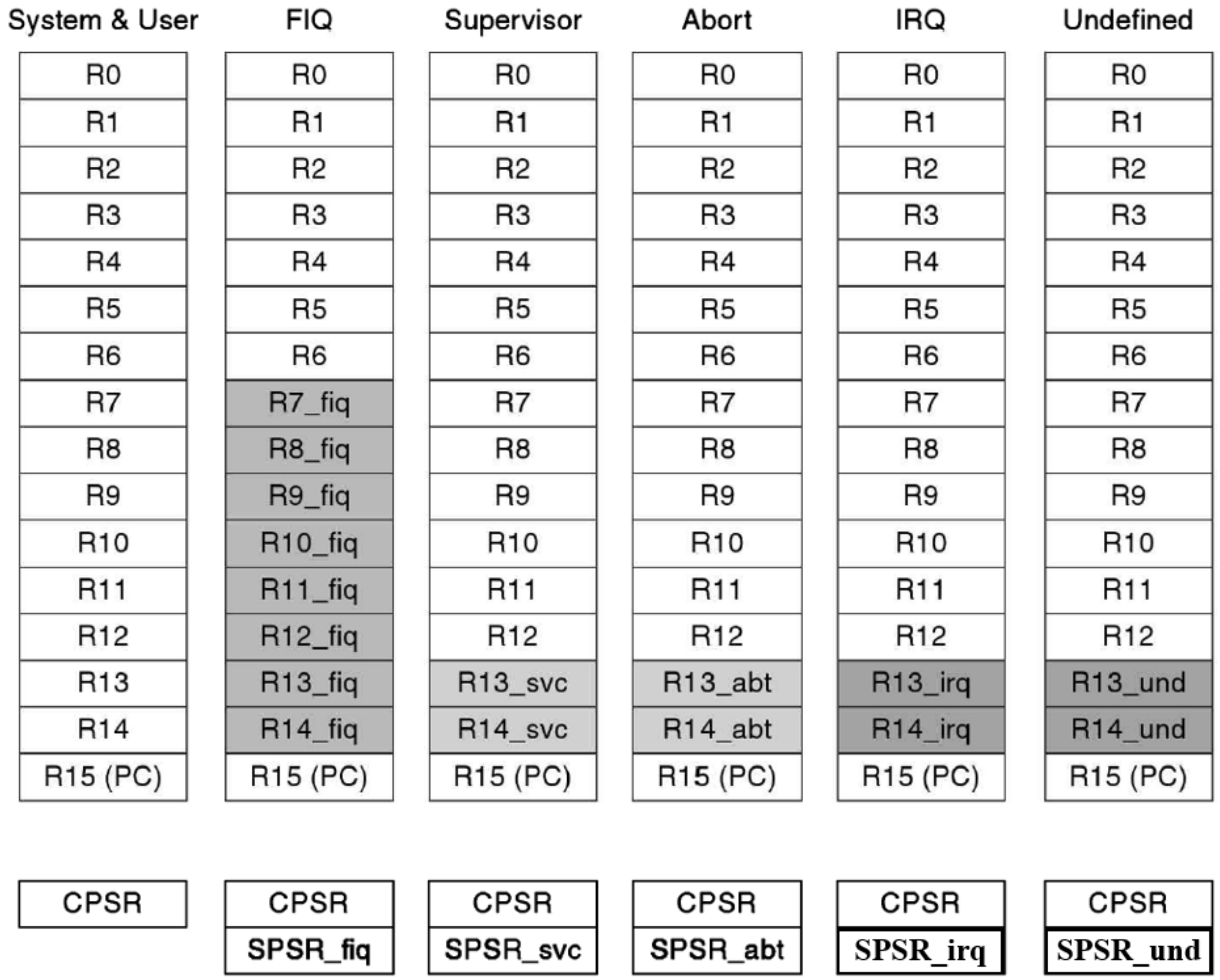
User registers R0–R6 are common to all operating modes.
when FIQ is entered →

R7–R14 replace user registers.

each modes have their own R13 and R14 =
each operating mode has its own unique Stack pointer and Link register.

ARM7 Architecture

Exception and interrupt modes →
Register set →



System & User	FIQ	Supervisor	Abort	IRQ	Undefined
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_fiq	R7	R7	R7	R7
R8	R8_fiq	R8	R8	R8	R8
R9	R9_fiq	R9	R9	R9	R9
R10	R10_fiq	R10	R10	R10	R10
R11	R11_fiq	R11	R11	R11	R11
R12	R12_fiq	R12	R12	R12	R12
R13	R13_fiq	R13_svc	R13_abt	R13_irq	R13_und
R14	R14_fiq	R14_svc	R14_abt	R14_irq	R14_und
R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_fiq	SPSR_svc	SPSR_abt	SPSR_irq	SPSR_und

ARM7 Architecture

Exception and interrupt modes →

Register set →

CPSR is common to all modes.
in each exception modes →
additional register SPSR (saved program status register) is added.

processor changes current value of CPSR after storing in SPSR →
this can be restored on exiting exception mode.

Exception modes →

entry to Exception modes is through interrupt vector table.
Exceptions are split into 3 distinct types →

1) Exception caused by executing instruction,
software interrupts,
undefined instruction exceptions,
memory abort exceptions.

2) Exception caused as side effect of instruction,
abort caused by trying to fetch data from invalid memory region.

ARM7 Architecture

Exception modes →

Exceptions are split into 3 distinct types →

3) Exceptions not related to instruction execution, reset, FIQ, IRQ interrupts.

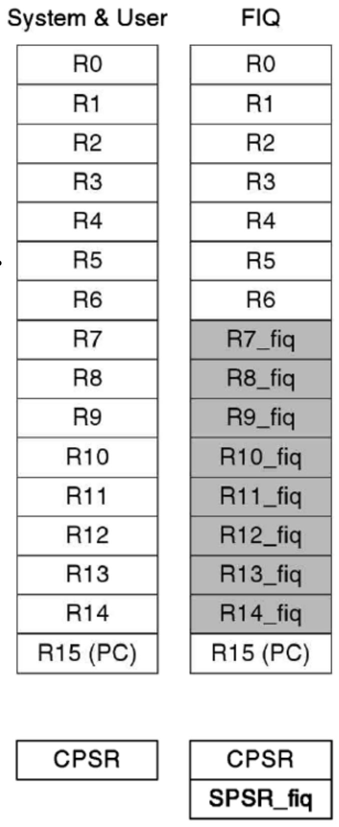
On generation of exception →

1) push some or all of user registers onto stack.
2) processor switched to privileged mode.
3) current value of PC is saved into Link register (R14) of privileged mode.
4) current value of CPSR is saved into privileged mode's SPSR.
5) IRQ interrupts are disabled.
6) if FIQ mode is entered, FIQ interrupts are also disabled.
7) Program Counter is forced to exception vector address.

processing of exception starts.

Supported data types →

1 byte, 2 bytes and 4 bytes,
signed and unsigned,
register and immediate data.


System & User
FIQ
R0
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15 (PC)
R0
R1
R2
R3
R4
R5
R6
R7_fiq
R8_fiq
R9_fiq
R10_fiq
R11_fiq
R12_fiq
R13_fiq
R14_fiq
R15 (PC)
CPSR
CPSR
SPSR_fiq