www.navneet.com
Page No.:
Date: 12 | July

Introduction of Embedded System

System : which components are required, how they worked, in which environmental conditions they works and gives the desired output.

embedded SIS : which performs the specific function as per the ~~desire~~ requirement

- perform specific task
- it does not need secondary memory as in a computer.
- Power saving, the software of the Es is pre-programmed and not changed by user

Classification :

(1) Based on Generation → (ASIC DSP)
(comp) first, second, third, forth gen^r

(2) Based on complexity & Performance requirement
Small scale
Medium "
large " / sophisticated

(3) Based on deterministic behaviour

(4) Based on triggering — event triggered
\ time triggered

Applications :

- Digital imaging
- Networking
- Digital TV
- Control & measurements
- embedded in wireless

Scanned by CamScanner

Embedded sys components :

- processor (GPP, SPP, ASSP, GPPs, COTs) (General Purpose, single, commercial off-the-shelf)
- Hardware (Timer, interrupt controller, I/O Devices, Memory, Ports)
- Software Program
- Operating sys (RTOS)

Classification

(1) Small Scale
- Single 8 bit or 16 bit Microcontroller
- need to limit power dissipation
- 8051 e.g.

(11) Medium scale
- Single or few 16 or 32 bit microcontrollers
- RISC
- Hardware & software complexity.

(iii) Sophisticated
- Vast soft & Hardware complexity
- scalable processor or configural processor and programmable logic arrays.
- we don't scaled down the size of processor we only scale down the func of it.

Components :

(A) Processor :

GPP :- device suitable for the variety of applications.
- general data path

Scanned by CamScanner

www.navneet.com
Page No.:
Date: 11 Aug

Technologies :

1. Processor Technology
2. IC "
3. Design "

1. Processor :

- Processor does not have to programmable
- it's not general purpose, Application specific but its single purpose processor.
- GPP (General Purpose Processor)
  - Program memory
  - Data Path (Register File and general ALU)
  - Data memory
- SPP (single Purpose Processor)
  - may or may not be programmable
  - size & Power is less
  - contain only components which required to execute a single program.
- Application - specific Processor (ASP)
  - prog. memory
  - optimized data Path
  - special func Units
  - Ex: FPGA (Prog. for ALU and comb on FPGA).

Scanned by CamScanner

www.navneet.com
Page No.:
Date: / /

- custom ALU
  (ALU restricted for particular application)

(2) IC technology :
- 12 to 15 layers
- gate level
- full custom, semicustom and PLD

(3) Design technology :

Synthesis, IP, Verification/testing.

Scanned by CamScanner

www.navneet.com
Page No.:
Date: / /

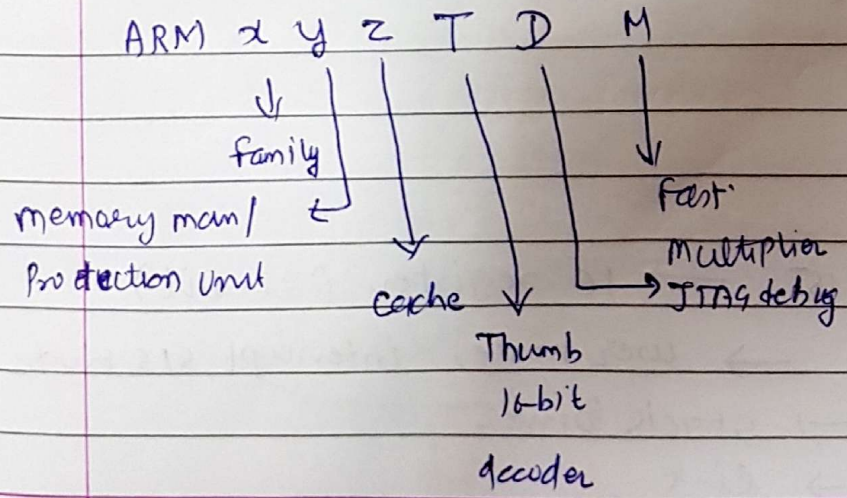
ARM PROCESSOR

- invention of RISC → ARPA's VLSI Project
  ↳ (advanced research Proj Agex
- ARM → acorn RISC machine
- low power and Peformance is good.

- RISC :
  - fixed and same length of instructions.
  - Pipelining
  - reduces the instructions
  - large general purpose register
  - load / store

ARM : ARM Processor has been specially designed to be small to reduce power consumption and extented battery operations.
- it's provides higher coding density
- ARM core is not a pure RISC, because of its primary application — the embedded SIs.

ARM nomenclature :



CPSR → SWI → software interrupt

Scanned by CamScanner

Cortex – A , M , R
↓ ↳ Real time
Application
Microcontroller

- Cache tightly coupled } → SRAM Memory
  ↳ For particular Application → memory addr are fixed (as 0 to 4 if required)

Barrel ~~Barrelled~~ shifter
- time reduce

1 [ ]
2 [ ]
4 [ ]
8 [ ]
shift

diff ways to attach the data → addr mode

Registers

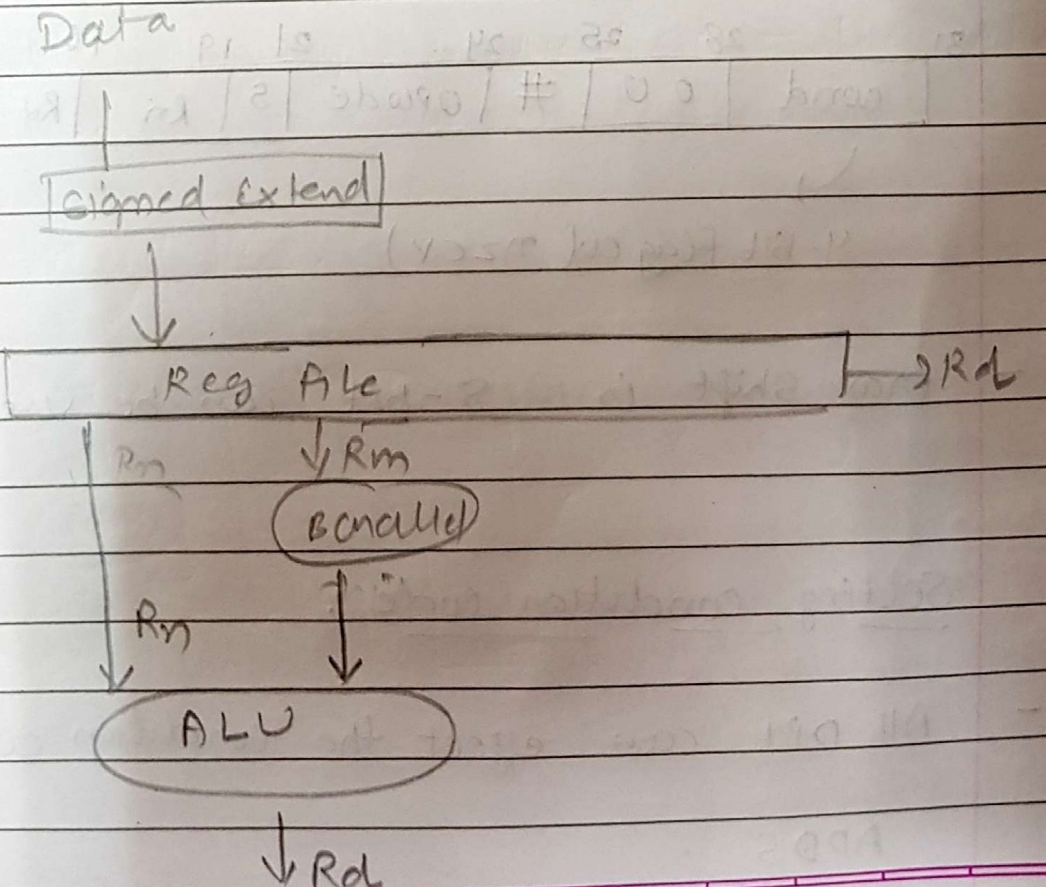
R0 – R15 → 16 register (32-bit)
R0-R12 → user mode, interrupt, SIS mode
R13 → stack Pointer
R14 → link
R15 → Prog. counter

Scanned by CamScanner

- CPSR
(current prog. status ~~Reegram~~ Register)
- Thumb mode
- Additional registe (SPSR) → (special prog status Rego)
- SPSR and cond in 32-bit registe both are updated by using 's'

adds

ARM Data flow



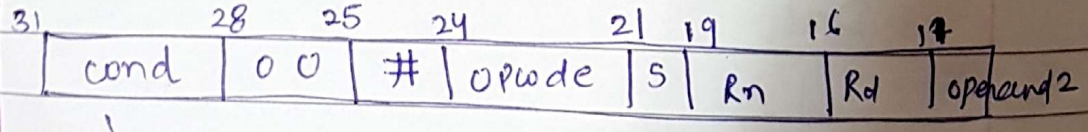
Scanned by CamScanner

Barrel Berallel shifter → shift Right / left all types of shifting operation is done by barrel shifter.

Types of Instructions:

(A) Data Bocessing Inst^r
- Register movement
- ALU
- Bit wise
- Compasion

* n z c v
↓ ↓ ↓ └→ overflow } flag
neg. zero carry



↳ 4-bit Flag out (nzcv)

- max. shift is 8-bit can be used.

Setting condition code:

- All DPI can affect the condition codes
ADDS.
↳ to upload the conditions on flag in

Scanned by CamScanner

Assembly Prog. / Language.

- CPSR

F I T

T → Set 0 bit for 16-bit taken
1 → 32-bit

F → fast interrupt
0 - enable
1 - disable

I → interrupt
0 ← enable
1 ← disable

⇒ 8 - Algo

Advanced Microcontroller Bus Architecture

AHB (advanced high Bus) ← Burst

Interrupt controller

APB (advanced Peripheral Bus)

Mem. controller

AHB & APB interface → using AHB APB Bridge

ARM7TDMI :

- Version 4
- Thumb
- Debug (on chip)

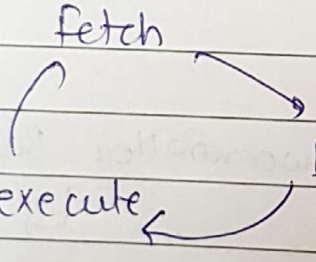
} TDMI

Scanned by CamScanner

- enhance multiply (DSP etc)
- Embedded ICE hardware
- Von-neumann arch.
  - 32 bit data bus
  - words - 4 byte
  - Half word - 2 byte
  - Data size - byte, word, half etc

RISC
- load & store arch. in memory
- R, I, J fixed length instruction format
- OPcode (destination, source 1, source 2)
- inline barrel shifter
- variable cycle instruction.

Instruction cycle :



ALU & other operation based on instr, destination etc.

- ARM supports both little endian and big endian memory format.
  0-15 mem. registers

Modes:

① User Mode
  R13 — Stack Pointer
  R14 — Link register
  (like return add store in RISC)

Scanned by CamScanner

www.navneet.com
Page No.:
Date: / /

for jump or branch instr
R15 → Prog. counter
CPSR → current prog. status Register

(2) Interrupt Request mode (IRQ)
R13, R14 & SPSR (save prog. status reg.)

(3) FIQ (fast interrupt Request) mode
R8 -14, SPSR (0-12) data, allows faster data tx than normal.
(for internal bus channel)

(4) Undefined mode (not in predefined format)
R13 -14, SPSR

(5) Abort mode
R13 -14, SPSR

- instruction fetch Abort — Data not Proper
fetch data — add of data memory
bcoz address not current

(6) (SVC) System mode - R13, 14 & SPSR
abnormal behaviours or first configuration
R0 - R7 → general Purpose

* CPSR:

flags	status	extension	control
N Z C V			(IFT Mode)

Scanned by CamScanner

IR → interrupt ~~mode~~ fast

I → 0 normal interrupt mode is enable

1 not enabled

F → 1 → disable

0 → enable

T → thumb mode (1-bit instruction)

0 → enable

1 → disable

mode → processor mode

SPSR → same arch. as CPSR

Exceptions:

priority ↑

① Reset (supervisor mode)

② software interrupt-(" ")

④ IRQ } ~~interrupt~~ mode

③ FIQ

② Data abort (abort mode)

⑥ Undefined instr

⑤ Prefetch abort (abort mode

→ fixed addr stored in IVT.

Arm Condition codes:

OPcode [31:28]		
0000	EQ	equal
0001	NE	not equal

Scanned by CamScanner

CS — carry set
CC — carry clear

- Register Movement operations:
  - MOV r0, r2 // r0 = r2
  - MOVN r0, r2 ; r0 = ! r2
    ↳ not equal
  - SUB r0, r1, r2 ; r0 = r1 - r2
  - RSB (reverse subtraction)

- Bit wise
  - EOR → ex-OR
  - BIC → r0 = r1 and (not) r2

- Comparison
  - CMP r1, r2 (compare) — flags set as a result of r1 - r2
    - r1 = r2 ; zero flag
    - r1 > r2 ;
    - r1 < r2 ;
  - CMN r1, r2 — r1 + r2 → flag set
  - TST r1, r2 — flag set as a result of r1 and r2 (bit wise)
  - TEQ r1, r2 — r1 XOR r2

- immediate operands
  - Add r3, r3, #1 ; r3 = r3 + 1
    #&FF
    ↳ max. (4 bit 2^4 = 16)

Scanned by CamScanner

- shift register Operands

ADD r3, r2, r1, LSL #3

↳ Logical shift left

; r3 = r2 + (r1 * 8)

By shifting left by 3

- Second operand must goes into the Barrel shifter

LSL - logical shift left RRX

LSR - " " Right (rotate

ASL - Airthmetic " left

ASR - " " Right

ROR - Rotate Right (used for swapping)

- LSL & ASL are same bcoz we are 0 extented into the LSB
- LSR → 0 extented into the MSB

ASR → MSB is same (bcoz of sign bit)

Multipication:

MUL r4, r3, r2

- immediate second operand not supported
- result register is not same as the first source

MULA r4, r3, r2, r1 → multiplying accumulation

r4 = (r3 * r2) + r4

Scanned by CamScanner

www.navneet.com
Page No.:
Date: / /

Ex 8- $r0 = r1 + (r1 * 4)$

add r0, r1, r1, LSL #2

Ex 9- $r2 = r3 * 119$

$= r3 * 17 * 7$

$= r3 * (16+1) * (8-1)$

Add r2, r3, r3, LSL #4 r3 r3x24

Add r2, r2, r2, LSL #3 r2 = r3(1+16)

→ Second operand is immediate value, that will have to convert as example.

- MULL which gives RdHi, RdLo = Rm * Rs
- MUAL

(when Rm, Rs 32bit