Sumedh

- Topics covered

1. Parallelism
2. Flynn's Classification
3. ARM features
4. Instruction coding
5. Basics I/O
6. ARM 5 / 7

- Parallelism

Executing 2 or more ops at the same time is known as parallelism

It's purpose is to improve computer system performance

Two or more ALUs or CPU can work concurrently

- Goals of parallelism :-

1. Inc computational speed
2. Reduce time
3. Inc throughput
4. Inc performance for a given clock speed
5. Solve bigger problems

- Applications :-

1. Socio economics
2. AI automation
3. Med apps
4. Weather app
5. Remote sensors
6. Genetic engg.

- Types :-

1. Hardware Parallelism
2. Software Parallelism

- **Hardware Parallelism**

  - Main Objective is to inc the processing speed

  - Enhances processing by using multiple processors, cores and threads

  - It's characterized by instruction issues per cycle :-
    1. K-cycle processor issues k instructions per sec
    2. Modern " " multiple instruc per cycle to handle more threads

  - Based on h/w there are two types :-

1. **Processor Parallelism** :-
   It means the comp arch has mutiple nodes, CPUs, socket, cores & threads

2. **Memory Parallelism**
   It means shared mmry, distributed mmry, hybrid distributed shared, multilvl pipelines, etc.

- **Software parallelism**

  - Defined by how code is written and optimized

  - It determines how much of the code can run in parallel

  - Factors lik algo design, prog style, optimization impact s/w parallelism

Type of s/w parallelism :-

1. Instruction -level
2. Task- level
3. Data- level
4. Transaction - level

- **Instruction - level parallelism :-**
  - ILP measures how many instruc can run in parallel
  - If instruc do not depend on each other they can run simultaneously
  - CPU designed to handle ILP process multiple instruc in one clock cycle

Consider the following example

1. x= a+b
2. y=c-d
3. z=x * y

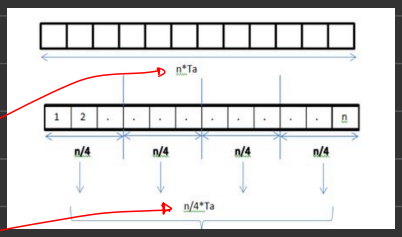
Operation 3 depends on the results of 1 & 2

So 'Z ' cannot be calculated until X & Y are calculated

But 1 & 2 do not depend on any other. So they can be computed simultaneously.

- **Data- level parallelism :-**
  - DLP involves parallelizing ops accross multiple data points
  - Focuses on distributing data
  - Operate concurrently on several data

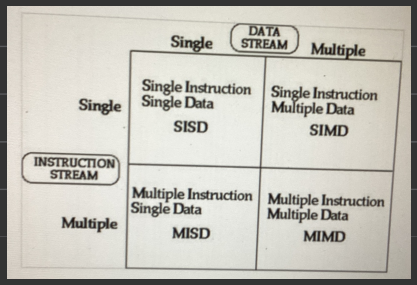
- Let us assume we want to sum all the elements of the given array of size n and the time for a single addition operation is Ta time units.
- In the case of sequential execution, the time taken by the process will be n*Ta time unit
- if we execute this job as a data parallel job on 4 processors the time taken would reduce to (n/4)*Ta + merging overhead time units.



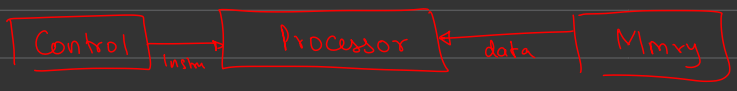
- **Flynn's Classification**

Flynn's taxonomy categorizes computer arch by how handle instruc streams (seq of cmds) and data streams (seq of data)

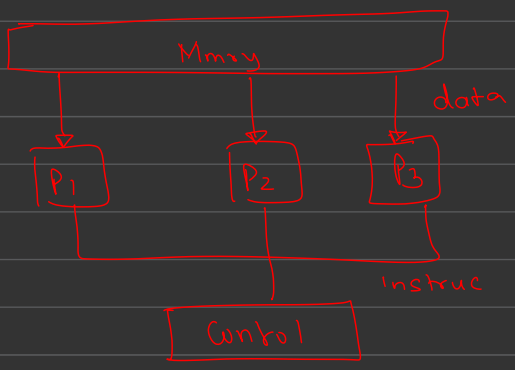
The four categories of flynn's Classification



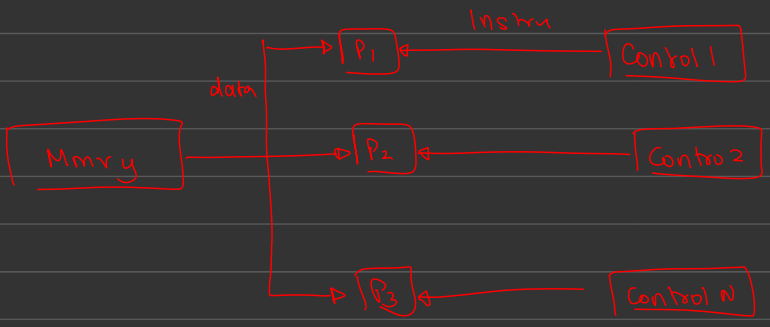
- **SISD (Single Instruc , single data)**
  - It processes one instruc & one data element at a time
  - Only 1 data / Instruc stream is acted by clock cycle
  - Deterministic execution
  - Typical in traditional , single-core comps



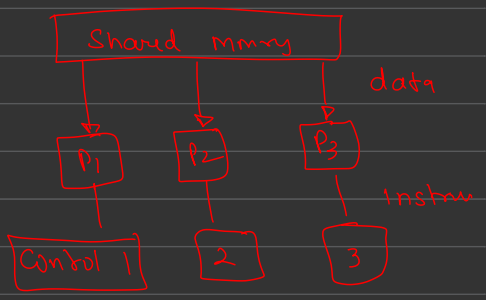
- **SIMD (Single Instruc , Multiple data)**
  - Here processor executes one instruc on multiple data elements simultaneously
  - They are connected by shared memory or interconnection
  - Common in system where same operation repeated on large data sets



- **MISD (Multiple Instruc , Single data)**
  - Each processor performs a different operation on the same data element
  - Same data flow thru a linear array executing diff ops
  - Useful in specilized apps



- MIMD (Multiple Instruc , Multiple data)
  - Each processor has it's own set of instruc & data
  - Can operate independantly
  - Can be sync / Async
  - Used in multi-core processors



- ARM features
  - ARM - Advanced RISC Machine
    - Reduced Instruction Set Computing
  - It's a 32-bit gen processor
  - High performance
  - low power consumption
  - Pipelining
  - Large reg file
  - Simple add modes
  - If 16-bit [THUMB Compressed Instruc set]
    - Half the size of ARM
    - Subset of ARM
    - less mmry power with more Instruc
    - For faster applications

- ARM registers
  - Gen-purpose regs either hold data or add
  - ARM has 3 special regs named r13, r14, r15
  - r13 : Stack ptr
    r14 : link reg
    r15 : PC
  - It has 18 active regs : 16 dat regs & 2 status reg

- **Processor modes**

  - These modes determine whether is privileged / non-privileged
  - Depending on that access is given
  - There are several modes

1. **Abort modes**
   The processor enters abort mode if failed access

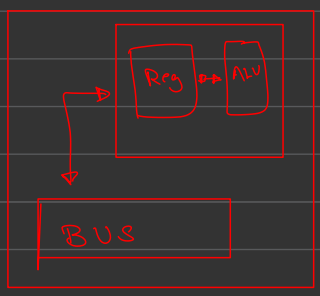
2. **Supervisor modes**
   The kernel system operates in this mode

3. **System Mode**
   It allows full read-write access to cpsr (reg)

4. **Undefined mode**
   Triggers in an undefined instruc

5. **User mode**
   Used for prog & apps

- **Core Arch**

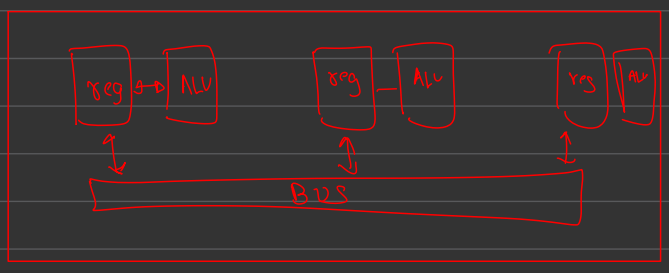


1. **Single Core :-**

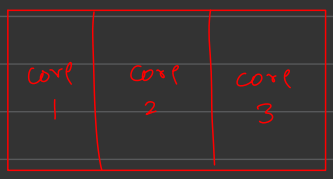
   - Trad. CPU with 1 core
   - Only handles 1 task at a time

2. **Multi-core**

   - Multiple core in 1 chip
   - Improved performance



3. Multi-core CPU Chip



- Parallel execution with cores
- Time-slicing (switches b/w thrds)

• Instruction Encoding

- Instructions are stored in mmry and fetched, decoded, and executed by the CPU

- There a Instruction Coding format :-
  - Opcode : Specifies the operation
  - Operands : Indicate the data

• MIPS

- MIPS (Microprocessor with Interlocked Pipeline Stage)
- It's a RISC arch with very few Instruc format
- The three formats :-

1. I-type (immediate)
2. R-type (reg)
3. J-type (jump)

• I-type (Immediate)

Used for Instruc that involve immediate values

eg ADDI $a0 , $12 , 33

↳ Add the value 33 to reg 12 & store in $a0

• R-type (Register)

Used for instructions involving arithmetic/logical ops on reg

eg SUB $7 , $8 , $9

↳ Subtracts the contents of $8 & $9 n store in $7

- J - type (Jump)
  - Used for jump ops where the prog control needs to move to diff add
  - Eg : Only 24 bits for add, so some taken frm PC

- Mmry load & store ops
  - The ARM is a load / store arch
  - Only these instruc can access mmry
  - Does not support mmry to mmry data processing
  - Types of Instruc
    1. LDR (Load) / STR (Store)
    2. LDM / STM : Stores multiple regs
    3. SWP : data swap blw mmry & reg

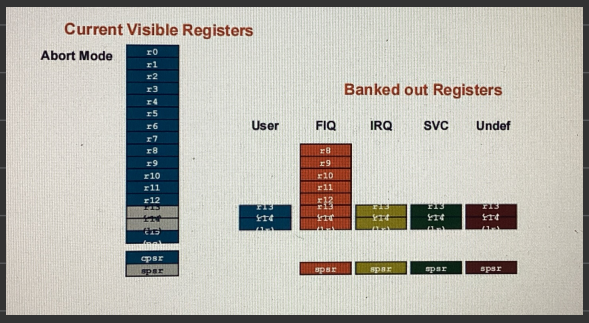
- Case Study of ARM 5 And ARM 7
  - Data Sizes in ARM
    1. Byte - 8 bits
    2. Halfword - 16 bits
    3. Word - 32 bits

32-bit ARM (word-aligned)
16-bit THUMB (half word - ')
8-bit Jazelle (byte code Java)

- ARM processor modes
  1. User - Normal tasks (unprivilege)
  2. FIQ - Fast Interrupt
  3. IRQ - Regular Interrupt
  4. Supervisor - Reset Interrupt
  5. Abort - Memry access violation
  6. Undef - undefined instruc
  7. System - Privileged

- ARM register set

Reg are 32 bil long & include :-
- Gen-purpose reg (r0 - r14)
- prog Countr (r15)
- Current prog status reg (cpsr)
- Saved " " " (spsr)
- ARM has 3 special regs named r13, r14, r15
- r13 : stack ptr
  r14 : link reg
  r15 : pc
- It has 18 active regs : 16 dat regs & 2 status reg



- Program status register (PSRs)

1. N Negative
2. Z Zero
3. C Carry
4. V Overflow
5. I IRQ disable
6. F FIQ disable
7. T Thumb State

- Condition Execution & flags

- ARM supports conditional execution
- Condition codes like EQ (Equal),
  NE (Not equal)
  GT (Greater Than)
- Flags (eg Z) are set using comparisons

- **Branch Instructions**
  - These are instruc that control the program flow
  - They perform arithmetic, logical, comparison & data movement

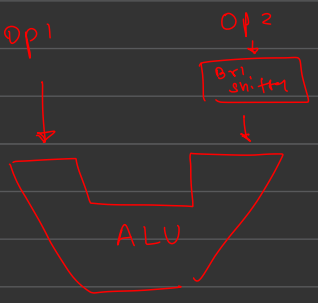
Arithmetic - ADD SUB
logic - AND ORR
Comparison - CMP TST
data - MOV MVN

- **The Barrel Shifter**

Adjusts operand 2 before an operation

Types of Shifts

1. LSL logical shift left (multiply by ^2)
2. ASR Arith shift rgt
3. LSR logical " rgt (divide by ^2)
4. ROR Rotate rgt
5. RRX " " with carry flag



- **Different ARM Ops**

1. LDM / STM
   - store / load multiple reg

modes

LDMIA / STMIA : Inc after
LDMIB / STMIB : " before
LDMDA / STMDA : dec after
LDMDB / STMDB : " before

2. SWI

↳ S/w interrupt request

Used by OS for secure tasks

Syntax : SWI {cond^n} <SWI no.>

3. PSR

modes

MRS / MSR : transfer status

↳ reads writes

- THUMB
  - It's a 16-bit compressed instruction set
  - Optimised for density code
  - Improved performance for narrow mmry
  - Switches from ARM using BX instruc

eg : ADD r2, r2, #1 32-bit

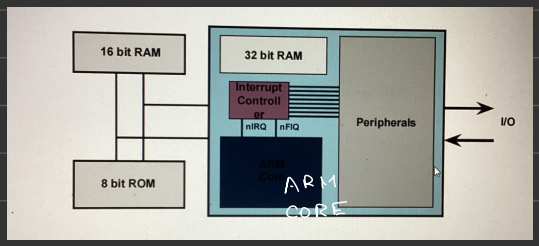
↓

ADD r2, #1 16-bit

- AMBA

↳ ARM based System

Components :-



1. RAM : 16-32 bit (main mry)
2. ROM : 8-bit (fixed data)
3. AMBA Bus : Connect peripheral
4. Interrupt Controller
5. Time I/O