06 - CE1006 - 15th Feb - Conditional Constructs

Branchless Logic (If Else)

- Avoid using conditional jumps as jump flushes the pipeline
- How?
    - Exploit Arithmetic relationship
        - AND & ROR
    - Conditional Execution
        - ARM Processor, MOVEQ

Switch Statement

- Sequential & Small Values
    - Keeping a Jump Table (JT) of all start addresses to case values
    - Move it into the Program Counter
    - MOV PC, [R2 + JT]
        - R2 = User Input
        - JT = Jump Table
- Random & Wide Values
    - Fork Algorithm - Divide and Conquer with nested ifs
        - It prevents the last case to be the worst case scenario

Loops

- Pre (While), Post(Do-while) tests
- While
    - Pre-test
    - Back CMP … <Inverse>
    - JLE Exit
    - {S0}
    - JMP Back
    - Exit
- Do-while
    - Post-test
    - {S0}
    - CMP … <No Inverse>
    - JGT Back
- For Loop
    - Pre-test
    - MOV [N], #0 <- Initialization
    - … {Same as While}
    - INC [N]
    - JMP Back

07 - CE1006 - 1st Feb

Shift & Rotate Operations
Shift
- Left or Right and will be pushed out to Carry bit
- 0 or 1 will be used to fill the bit

Rotate (Cyclical shit) [ROL, ROR]
- Shifts LSB to MSB or MSB to LSB
- Bit will also be stored into Carry

Shift
- performs Multiplication (shift left) & Division (shift right)
- each shift is 2^n
  - e.g. shift left by 2. 2^2 = 4

BCSR - (Bit Clear Status Register)
- Clears a particular Status Register flag

Signed Divide using Arithmetic Shift Right [RAR]
- Shift MSB Value to the next MSB (Keeps Sign Flag)

Rotate Counter [RCN]
- Sets the counter to Rotate [ROR, RAR]

Rotate Left with Carry (RLC)
- Push Carry into LSB

Program Control Instructions
Conditional Jump (Jcc)
- JEQ (JZ). Jump = 0
  - Jump when result = 0. * Note: PC is ahead on next instruction
  - Use labels instead.

Module Programming
Subroutine Implementation (CALL and RET)
- Writing as Modules (Libraries)
- Designed and tested Independently
- Reduce Program Size
- Reuse in other projects

CALL - Subroutine call
- Similar to JMP with a Return

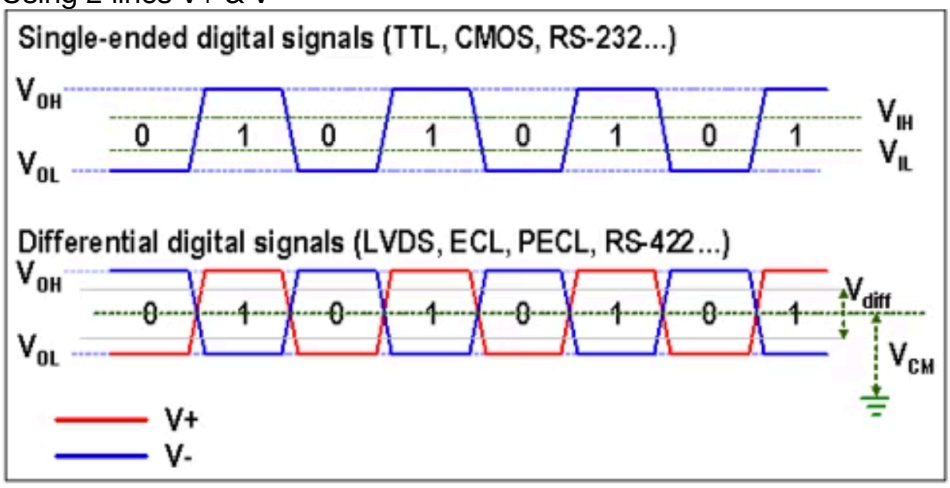
07 - CE1006 - E-Learning - 23rd Feb

Analog Signal

- Real world signals are Analog in nature
- Continuous Time and Waveform
- **Analog-to-Digital Converter (ADC):** Converts analog to digital value
- **Digital-to-Analog Converter(DAC):** converts processed signals back to Analog
- **Filter:** Smooths out DAC Signals

Interface Compatibility

- Communication Protocol
- Electrical signal level
    - Similar voltage levels between two devices
        - Transmitting Device: V(OL) [0] & V(OH) [1]
        - Receiving Device: V(IL) [0] & V(IH) [1]
        - V(OH) >= V(IH)
        - V(OL) <= V(IL)
    - Differential Signals
        - Better noise tolerance to allow higher clock frequency
        - Using 2 lines V+ & V-



Parallel Data Transfer

- Prone to **Signal Skew** and **Crosstalk**
- Design is simpler as Strobe Signals are needed
    - Data is being sent when Strobe Signal is asserted
- **Signal Skew**
    - data lines takes different amount of time to reach the receiver causing the wrong data to be latched
    - variation in **propagation delay**
        - contributed mainly by Capacitance and Resistance. e.g. Variations
            - T = RC - Larger RC, Longer Delay
- **Crosstalk**

    - Interference from neighbouring signals
    - **More noise** = limits speed
- **Front-Side Bus (FSB)**
    - FSB to interface with CPU
    - has a bottleneck if connected with multiple connections

Serial Data Transfer

- **Advantage**
    - Less affected by Signal Skew & Crosstalk
- **Disadvantage**
    - Interface design is more complex as it need to handle serial to parallel conversion
- **Data Transfer**
    - Simplex - Single direction
    - Half Duplex - Both direction but only one active
    - Full Duplex - Simultaneous RX & TX
- **Synchronous vs Asynchronous**
    - Synchronous
        - Common clock signal
        - Master-Slave config: Master providing Clk
        - Potential faster transfer as no data overhead to synchronous data
        - **Inter-Integrated Circuit (I2C) Bus**
            - Half Duplex
            - Two wire bus (Multi-Master, Multi-Slave)
                - SDA: Data
                - SCL: Clock
            - Slave & Master can transmit
            - Master sets the Clock
            - Start Sequence
                - SCL: HIGH
                - SDA: Pulled LOW
            - End Sequence
                - SCL: HIGH
                - SDA: Pulled HIGH
            - Data is changed when SCL is LOW
            - Data is latched at rising edge of SCL
        - **Serial Peripheral Interface (SPI) Bus**
            - Full Duplex
            - 1 SPI Master, Multiple Slaves
            - Multiple Slave Selects on Master
            - Data Transfer: Slave Select Pulled LOW
            - MOSI: Master-Out(put)-Slave-In(put)
            - MISO: Master-In(put)-Slave-Out(put)

Asynchronous Transfer

- No common clk
- Special SYNC to indicate START/STOP

- Potential Skew as it progresses
    - Due to differences in clock
    - SYNC words/buts provide timestamp to resynchronize
- Universal Asynchronous Receiver Transmitter (UART)
    - RS232
    -

LOGIC LEVELS			
STANDARD		LOGIC 0 (Volts)	LOGIC 1 (Volts)
TTL	INPUT OUTPUT	0.0 to 0.8 0.0 to 0.4	2.0 to 5.0 2.8 to 5.0
RS-232	INPUT OUTPUT	+3 to +15 +5 to +15	-3 to -15 -5 to -15

- UART Transmit
    - No Clk involved
    - START Patten (login ‘0’) - Falling Edge
    - DATA sent out at a certain **Baud Rate**
    - STOP pattern (1 or more bits) terminates transmission
- UART Receive
    - Falling Edge trigger interrupt for data transfer
    - Receiver Clk usually run must faster.
        - e.g. 16x of Baud Rate of Transmitter to time the sampling close to the middle of each data bit
        - Oversampling increases accuracy
            - Taking more samples close to the middle of each data bit
- Universal Serial Bus (USB)
    - Grounded Shielding, Twisted pari reduce effect of Crosstalk
    - NRZI (Non-Return-to-Zero Inverted)
    - Bit Stuffing
    - Full Duplex in USB 3.0

Summary

- Interfacing Consideration
    - Electrical Signal level
    - Communication Protocol
- Types of Signal
    - Analog
    - Digital
- Digital Interface
    - Parallel
    - Serial
- Interface Standards
    - INTEL FSB
    - IEEE 488 (GPIB)
    - UART
    - USB

08 - CE1006 - 7th Mar & 10th Mar

Data Transfer

- Polling Technique - Programmed I/O
    - CPU polls certain IO port
    - CPU has full controlled dedicate 100% resources
    - Watching a bowl of water bowl
    - Advantages
        - Minimum hardware needed, easy to debug
        - Programmer has complete Control
    - Disadvantages
        - CPU stays in a loop, other tasks don't get done
        - Program execution held up waiting for device to be ready
- Interrupt
    - Signal notifies CPU when event occurs
    - CPU suspends its current program to service the interrupt
    - Interrupt Service Routine (ISR)
        - A short program to process the interrupt
    - Waiting to hear the whistle of water boiling
    - Advantages
        - Efficient use of CPU as it doesn't need to monitor I/O devices
        - Allows **prioritisation** and **pre-emption**
    - Disadvantages
        - More hardware interface which is complex and difficult to debug
- DMA Controller
    - Relieve CPU of data transfer task
    - Config
        - Src Addr
        - Dest Addr
        - Amt of data to transfer
        - DMA Trigger signal
            - Allow DMA to take over system bus
    - Design
        - Fetch-n-Deposit
            - Internal buffer to manage data transfer
        - FlyBy
            - Data transferred directly from src to dest
            - Additional Address line in DMAC
    - Modes of Transfer
        - Burst
            - DMA takes control
            - returns CPU control until transfer is completed
            - Used for high data transfer
        - Cycle Stealing
            - Releases data bus after one word transfer
            - Executed between
                - CPU Instructions
                - Pipeline stages
            - Used for responsive (real-time) systems
        - Transparent

        - DMAC transfer when system bus is not in use by CPU
        - CPU is not affected

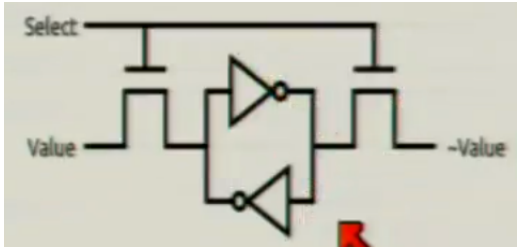
10th Mar
Computer Memory

- Semiconductor Based
    - SSD, SD Card
- Magnetic Based
    - Magnetic
- Optical Based
    - CD, DVD

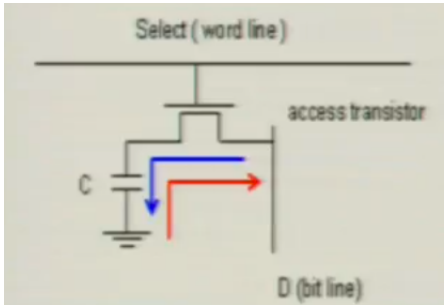
Volatile & Non-volatile

- Volatile
    - Data lost when power is removed
    - SRAM & DRAM
- Non-volative
    - Data retained when power is removed
    - Magnetic storage

RAM

- Static Ram
    - Data stored as long as supply is applied
    - Access
        - Addr
        - Data - Data bus
        - CS - Chip Select
        - WE - Write
        - OE - Read
    - Access via
        - Word lines
            - Horizontal
            - Controls 8 bits
        - Bit lines
            - Activates the line onto the data bus
    - SRAM Cell
        - 

- Dynamic RAM
    - Periodic refresh required
        - Stored charge leaks

    - Uses a Capacitor
    - Read process destroys information
    - DRAM Cell
        - 

    - Currently using Synchronous SDRAM
        - Increase speed using synchronous clock

EPROM & EEPROM

- EPROM
    - Erasable Programmable ROM
    - Put under UV light to erase program
- EEPROM
    - Electrically Erasable Programmable ROM
    - Electrically program or erase
- MOSFET
    - Works like a switch
- Floating Gate Transistor
    - Similar to MOSFET
    - Extra layer (Floating Gate)
    - Transistor Off when programmed (contains charges)
        - Logic 0
    - Transistor On when Erased (no charges)
        - Logic 1

NAND & NOR Flash

- Like EEPROM
- Erased in larger block sizes
- Erase cycle is slower than Read/Write
- NOR Flash
    - Support Execute in place
- NAND Flash
    - Does not support execute in place

09 - CE1006 - 11th Feb - Modular Programming

Calling Subroutine

- Pushing Parameter into Stack
- Using ADD SP, **#2** (Pops two parameter pushed into the stack)

Transparent Subroutines

- Saving away all registers before subroutine for it to prevent overwriting registers in main program
- PSHM 0x00C (1100) <- (R3,R2,R1,R0)
- POPM to restore registers

Accessing Stack Parameters

- Finding offset from Pushing to Calling using Stack Pointer (SP)
- Note: CALL puts Return Program Counter (PC) Value into the stack
- PSHM pushes register values to save into the stack

Passing by Value or Reference

- Reference: Arrays, Modifying variable in subroutines
- Value: PSH #5
- Reference: PSH #0x100

Local Variables

- Loose coupling (Local variables) - Data within module is entirely independent of other modules
- Stack Frame: temporary memory space
    - created right after Return Address
    - Can be accessed using **Frame Pointer (FP)** - Start SP Address of subroutine

Using the Frame Pointer

- Must R3 be saved to use as Frame Pointer?
- Stack Pointer can be used
    - Pro: Efficient without the setup of Frame Pointer
    - Con: More restrictive

Summary

- CALL, RET: Basic instruct for subroutines
- Stack: multiple parameter passing
- Transparent Subroutines with PSHM, POPM
- Local Variables in subroutine using a **Stack Frame**

Control Flow Constructs

If Statements

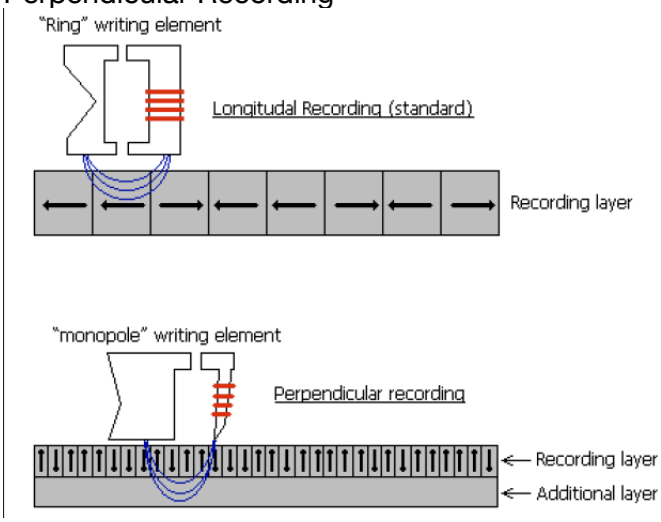
- In more than condition, find the inverse for a more efficient code
- Using less than and equal
    - CMP a, b
    - JLE skip <- Jump if less than equal
    - {If code}
- If-Else
    - Jump to the Else situation
    - Note: **JMP required** when {If code} is completed
        - JNE DoElse
        - {If Code}
        - JMP skip
        - Do Else {Else Code}

Compounded Conditions

- AND (&&) Conditions
    - C Code: if ((a == b) && (b > 0)) {S1}
    - Low Level:
        - if (a != b) then Skip
        - if (b <= 0) then Skip
        - {S1}
        - Skip ...
    - Note: Keeping the least likely condition leftmost makes the program more efficient
- OR (||) Conditions
    - C Code: if ((a == b) || (b > 0)) {S1}
    - Low Level:
        - if (a == b) then DoIf
        - if (b <u>**<=**</u> 0) then Skip **<- Reversed Condition**
        - DoIf {S1}
        - Skip ...

09 - CE1006 - Magnetic Hard Disk

Types of Recording

- Longitudinal Recording
- Perpendicular Recording
- 

Dissection of Magnetic HDD

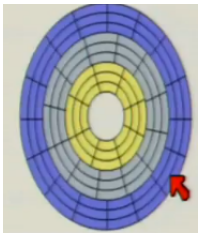
- Platters
- Tracks
- Sectors

HDD Transfer Rate

- Seek Time T_S
  - Time Taken for head to move to correct track
- Rotational delay T_R
  - Time taken for disk to rotate till start of target sector
  - Revolution Per Minute (RPM) usually converted to RPS
  - RPM / 60 = RPS
  - T_R = 0.5 / RPS seconds
- Access Time T_A
  - Time from request till head in position
  - **$T_A = T_S + T_R$**
- Transfer Time T_T
  - Time to transfer required data after head is positioned
  - **$T_T = N / RPS * D_T * D_S$**
  - N = Number of bytes to read
  - D_T = Number of sectors per track
  - D_S = number of bytes per sector
- Total Time

    - $T_{Total} = T_A + T_T$
- It is notable that Access time is much higher than Transfer time thus the slow speed when hard disk is fragmented

Physical Layout - Hardware View

- Tracks divided into zone, with different number of sectors / track
- 

Logical Layout - Software View

- Cylinder-Head-Sector (CHS)
    - Legacy Scheme due to unable to meet current space standards
- Logical Block Addressing (LBA)
    - 48 bit allowing up to 128 PetaByte

Datasheet

- Complexity of Hard disk is expanding
- Most useful resource are
    - Bytes per Sector
    - Number of Sectors
    - Total Logical Data Bytes

Solid State Drive (SSD)

- Based on NAND-Flash or NOR-Flash
- Limited Program-erase cycles (3,000 - 1,000,000)
- Extending Techniques
    - Wear Leveling
    - External RAM as buffer
- Mean time between failures (MTBF) for SSD is comparable to HDD

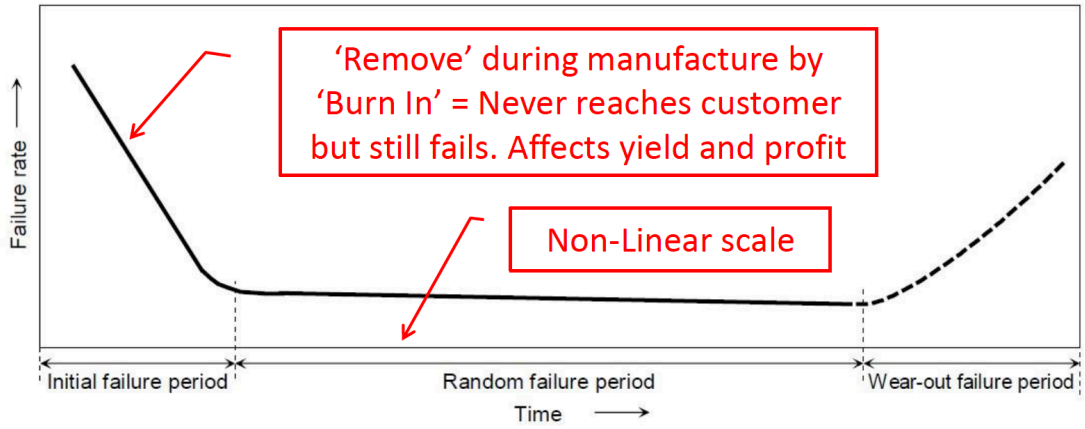
HDD vs SSD

- HDD
    - Advantages
        - Lower Cost per Bit
        - Almost infinite erase cycle
    - Cons
        - more prone to crashing if HDD is dropped or shaken
        - Slower Transfer Rate
- SSD

    - Advantages
        - Lighter and occupy less space
        - High Transfer Rate
    - Disadvantages
        - More Costly compared to HDD
        - Finite Erase Cycles

Reliability – Bath Tub Curve

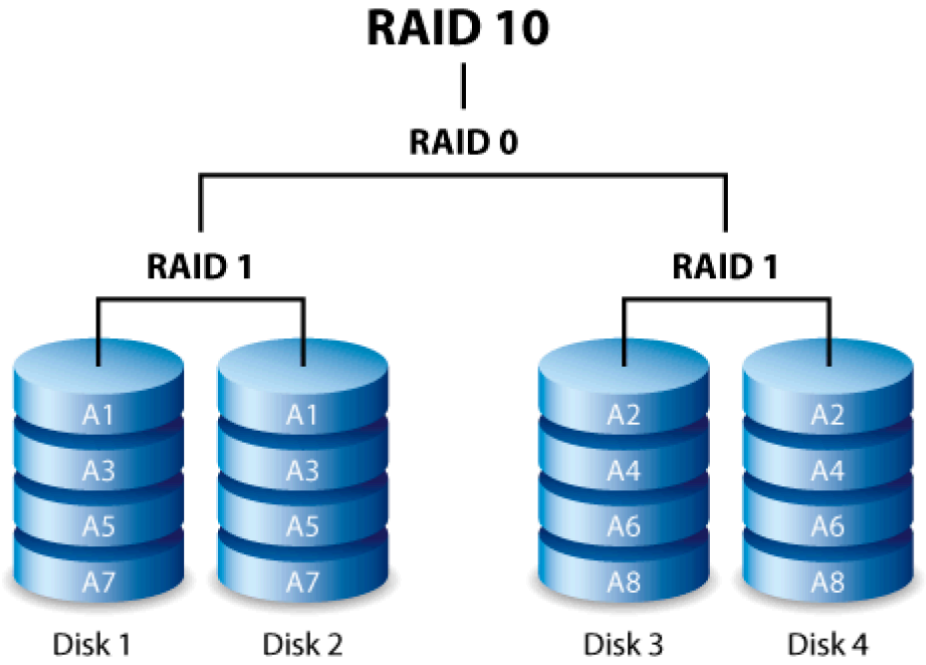
- Initial Failue
- Random Failure
- Wear-out Failure



-
- Burn in - Stress a product
- MTTF - Mean (Average) Time To Failure
- $R(t) = e^{-\frac{time}{MTTF}}$
- $R(t) = e^{-\frac{MTTF}{MTTF}} = e^{-1} = 0.368$

Redundancy and RAID

- Redundant Array of Independent Disks
- RAID 0
    - Performance focused
    - No redundancy
- RAID 1
    - Mirrors Hard disk
    - Faster Read Operations
    - Write throughput is slower
- RAID 5
    - Higher performance
    - Stripping of file with distributed parity (Error Correction)
    - RAID 3 - 4 Similar to 5

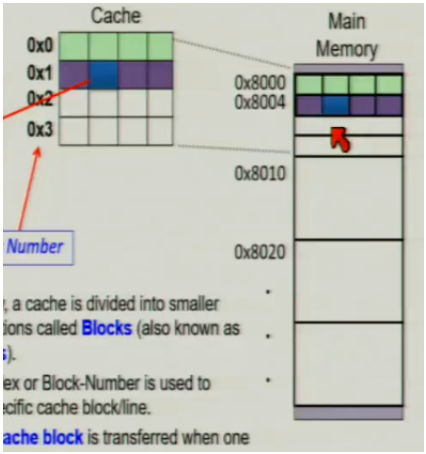
- RAID 6
  - Recover from multiple disk failures
  - Double distributed parity
- RAID 10 (Not Official)
  - Combination of RAID 1 + RAID 0
  - 

09 - CE1006 - System Memory

Cache

- Buffer between CPU & Main Memory
- Ghz Speed
- External Memory 100s Mhz
- speed up recently used data
- Common Data Mapping Scheme

CPU Memory Access

- CPU sends request to Cache
- If not in cache, request from Main Memory, if not from Disk
- Once located, required data and **nearby data elements** are fetched into cache
- whole cache block is transferred when one or more bytes required
- 

<u>Common Data Mapping Scheme</u>

- **Direct Mapped Cache**
  - One block can only go into one cache block location
  - Corresponding **Tag Value** keeps track of the main memory block
  - Tag, Block, Offset
  - 

  - Advantages
    - Simple
    - Found easily and rapidly
  - Disadvantages
    - Prone to Cache Trashing
- **N-way Set Associative Cache**
  - **e.g. 2-Way** Set Associative Cache
  - N Blocks forms a Set, 2 Blocks forms a set
  - Tag, Set, Offset
  - Advantages
    - Less Cache Trashing

    - Disadvantage
        - Longer search time
        - Complex Design
- **Fully Associative Cache**
    - Data can be in any Cache Block
    - Tag, Offset
    - Extreme case of N-Way Set Associative Cache

Principles of Cache Design

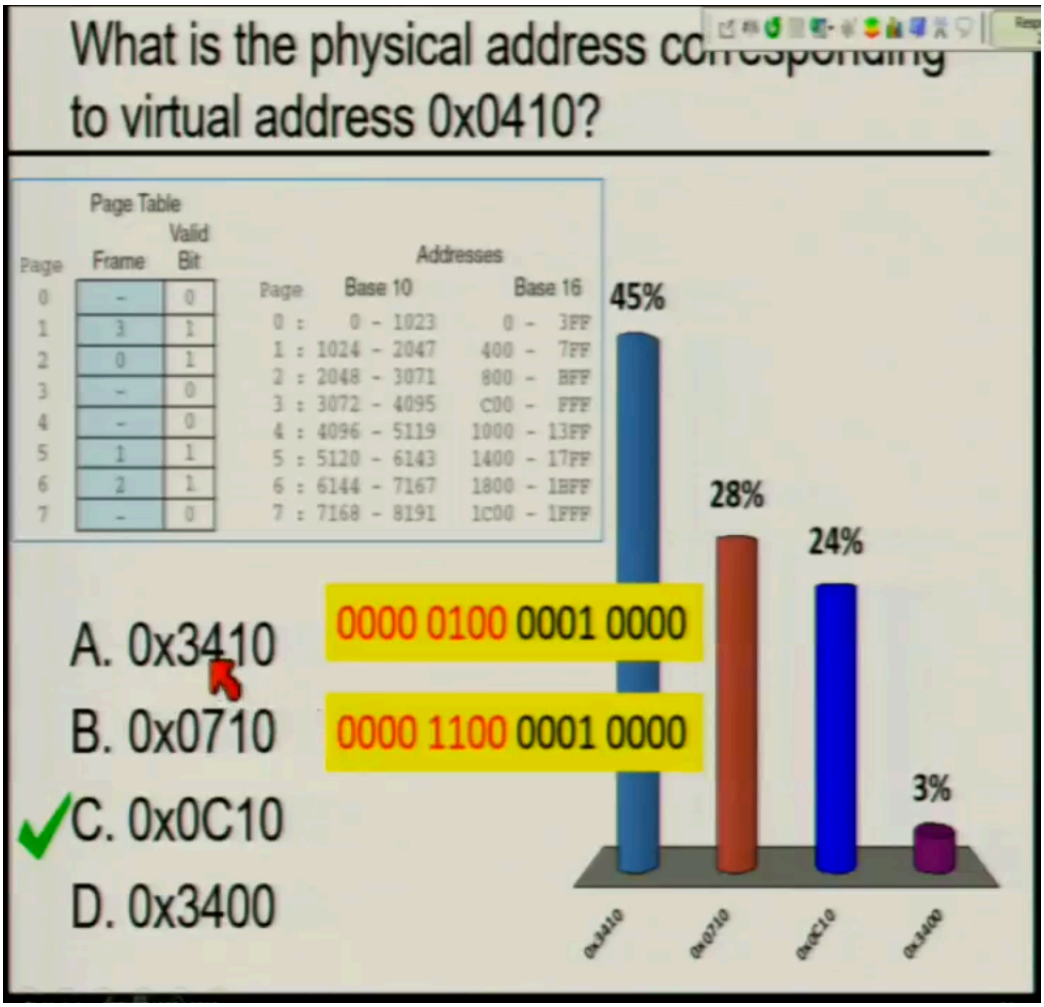
- **Storage**
    - Principle of locality
        - Due to array usage, chances of using a **nearby data element** accessed is high
        - Locality of Space (Spatial Locality)
            - Code/Data close together
        - Locality of Time (Temporal Locality)
            - Recently executed code
    - Cache Memory Replacement Policy
        - Least Recently Used (LRU) Algorithm
        - First-in, First-out (FIFO)
        - Random
    - Cache Write Policy
        - Dirty Blocks
            - Bytes changed in cache
        - Write Through
            - updates cache and main memory simultaneously
        - Write Back
            - Updates memory only when block is to be replaced
    - Write Miss Policy
        - Write Allocate??
        - Write-no-allocate
            - Write directly to main memory
- Retrieval
    - Cache Hit - Data in Cache
    - Cache Miss - Data not in Cache
    - Cache Hit Rate - Percentage of time data in cache
    - Cache Miss Rate - Percentage of time data not in cache
    - Miss Rate = 1 - Hit Rate
    - Hit Time - Time to access cache memory
    - Miss penalty - Time to process a miss
    - Effective Access Time
        - $EAT = H \times Access_C + (1-H) \times Miss\ Penalty$
        - $EAT = H \times Access_C + (1-H) \times (Access_C + Access_{MM})$
    -

09 - CE1006 - Virtual Memory Management

Virtual Memory

- Allows programs bigger than size of main memory to be processed
- Memory Management Unit (MMU)
    - Hardware Device that does conversion
- Paging
- Segmentation
- Advantages
    - Frees application from managing shared memory space
    - Virtual Address Space to appear contiguous on fragmented blocks

Paging - Not Understood

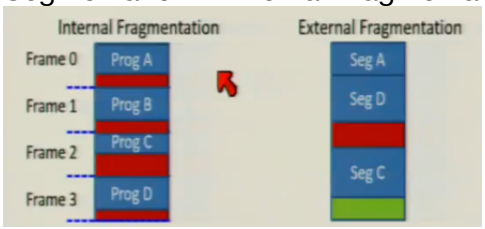
- Page Table to Convert
- Frame - Physical Page numbers in main memory
- Page - Virtual Page numbers in program code
- Page Fault - Data not found in main memory
- Reference to only Valid Bit 1
- Offsets remain the SAME
- 

- Translation Lookaside Buffer (TLB)
    - Page Table Cache that stores recently accessed values

Segmentation Method

- Variable-length segments
- entries in a **Segment Table**
- Note: Still **Page Fault**, not segmentation fault

Memory Fragmentation

- Memory too small to contain any data and rendered useless
- Paging - Internal Fragmentation
- Segmentation - External Fragmentation
- 

11 - CE1006 - 18th Feb - Data Transfer and I/O Interfaces

Lab 3 - UART Communication

Interface

- Boundary where two or more devices exchange data
    - Networks: Peer-To-Peer or Multi-point
    - Data Transfer: Serial(Single Bit) or Parallel(Multiple Bits)
    - Single or Bi Direction

Serial Interface

- Transfer Modes
    - Simplex - One Sender, Receiver
    - Half Duplex - One Line, Sender/Receiver
        - e.g. Walkie Talkie
    - Full Duplex - Two Lines for TX & RX
        - e.g. - Handphone

Asynchronous vs Synchronous

- Asynchronous
    - No Common Clock
    - Receiver needs to know transmitting clock rate
    - Special SYNC words to track START/STOP
- UART - Universal asynchronous receiver/transmitter
    - Baud Rate - Transmitting clock rate
    - START pattern
    - PARITY bit (optional)
    - STOP pattern

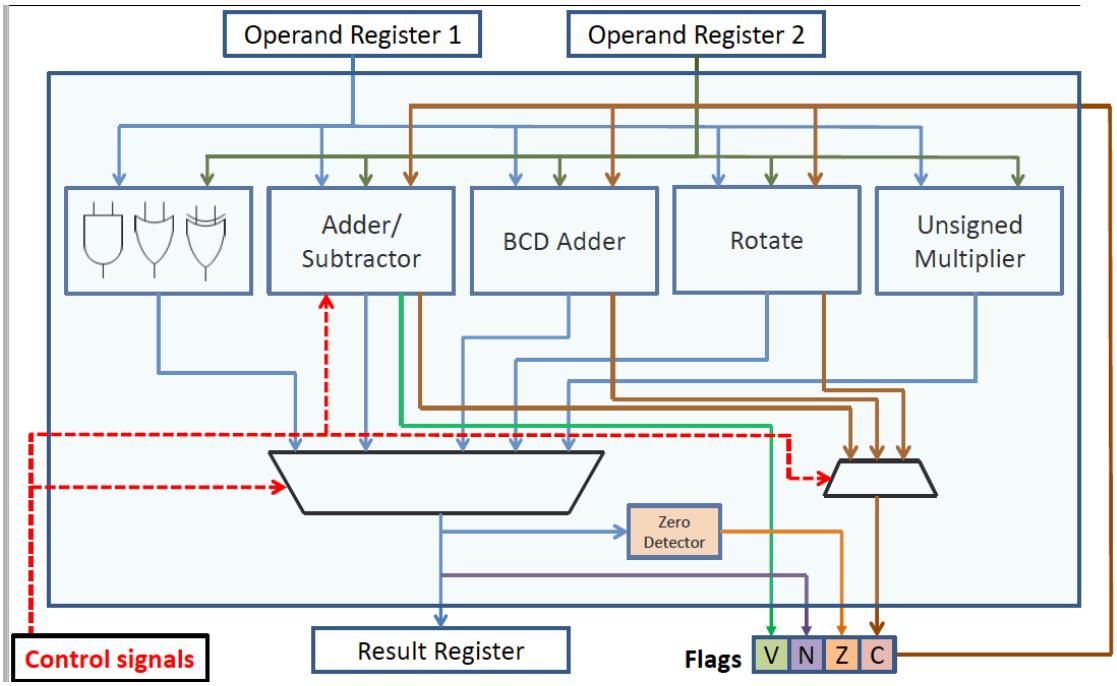
Parity Bit

- Even Parity
    - 010**1** - Parity Bit is one to make all even 1 bits

11 - CE1006 - Arithmetic and Logic Unit

Arithmetic and Logic Unit

- Arithmetic - Addition, Subtraction
- Logic - NOT, OR ...

- ALU of VIP Processor
- 

Positional Numbering System

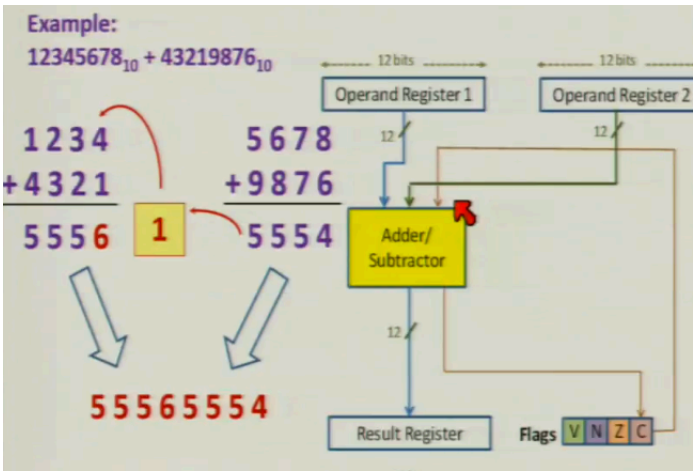
- As number gets larger, the benefits increases

Decimal to Binary Conversion by Subtraction

Signed Magnitude vs 2's complement

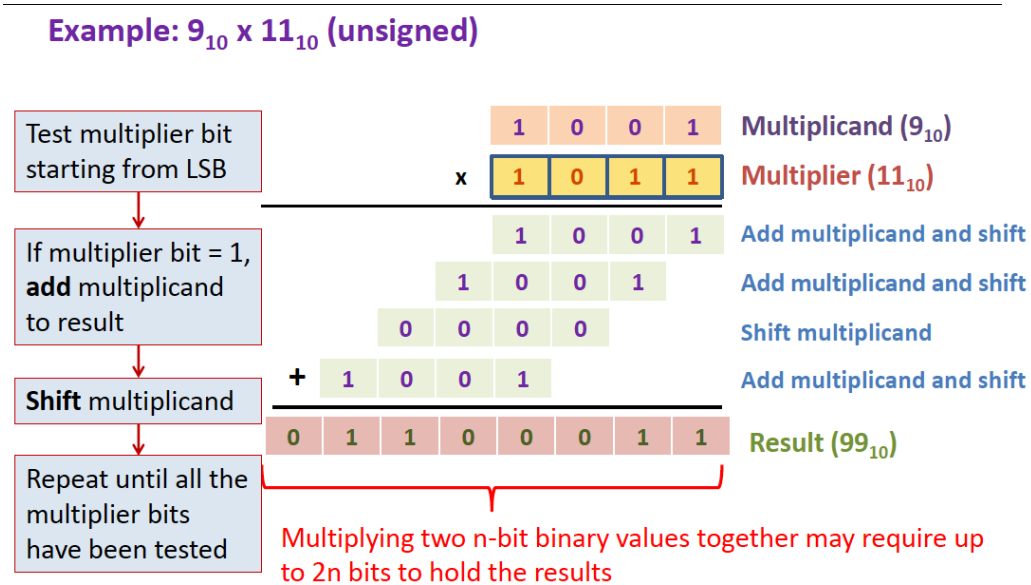
- Unsigned Magnitude
  - Overflow when carry is 1
- Signed Magnitude
  - 0111 = 8
  - 1111 = -8
  - Easy for people to understand
  - Requires complicated hardware
  - Two representations of zero
- Two's Complement
  - Detecting Overflow
    - Same sign and result sign in inverted sign
    - Carry does not determine overflow

Multi-Precision Arithmetic

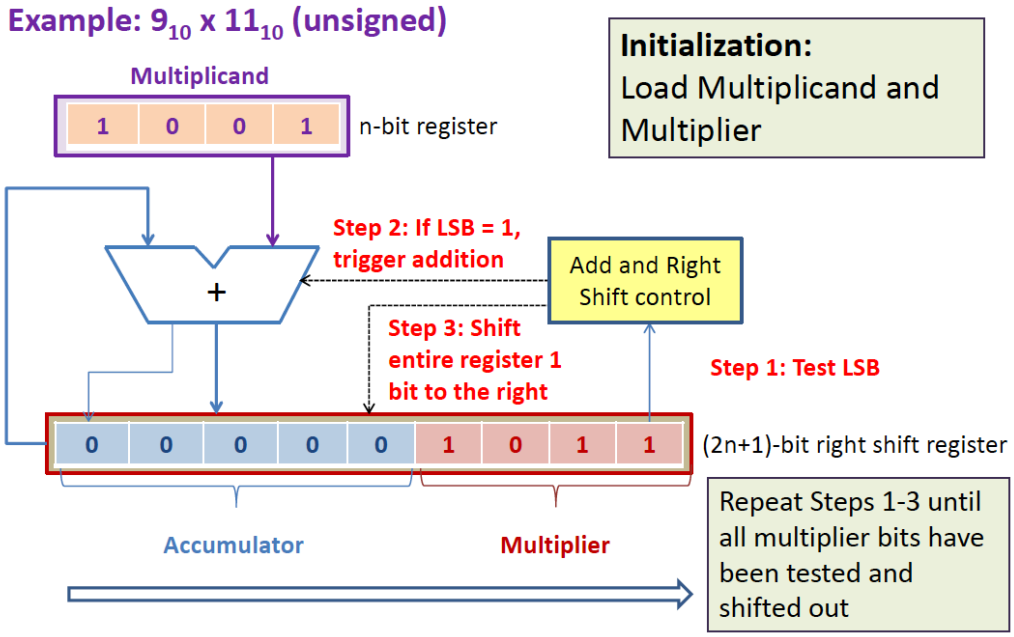
- 

Binary Multiplication

- Successive Addition
  - 4 * 3 = 4 + 4 + 4
- Partial Product



  - Multiplier, Multiplicand, Result, Number of Multiplier bits
  - UMUL <Multiplier> in VIP
  - Multiplicand in R0
- Hardware Multiplier



- Booth's Algorithm
  - Optimise number of addition cases

Current Bit	Preceding Bit	Booth's Form
1	0	-1
0	1	+1
0	0	0
1	1	0

  - Add virtual 0 behind LSB

Understand Signed Multiplication
Multiply Negative Numbers

11 - CE1006 - Floating Point Unit

Range

- Smallest and largest representable number
- Each tick represents a number in the range

Precision

- Amount of information to represent each number
- Number of ticks represent the precision

Radix Point

- dot to represent the point between Range and Precision

Number of bits are the same

- Precision = 1 / Range
- Range increase, Precision Decreases

Floating Point Representation

- Sign
  - Sign bit
- Mantissa
  - Base Value
  - Size determine precision of values
- Exponent
  - Specifies decimal of radix point
  - Size determine range of representable numbers
- Small Numbers require high precision, Large numbers require lesser precision

Normalisation

- Single integer cannot be zero

Underflow

- When number is too small to be represented

Guard Bit and Rounding

- Guard Bit
  - When add/sub exponents are required to be aligned and some bits might be lost
  - Maintain precision during computation
- Rounding

    - Remove the Guard Bit by rounding
- Rounding Error
    - Results are inaccurate due to limited bits
    - Adding a list of very small numbers
    - Workarounds
        - Add/Sub similar exponent
        - Start with smaller exponents

Multiplication

- Multiply Mantassa
- Add Exponents
- Normalise if required

IEEE 754 Floating Point Standard

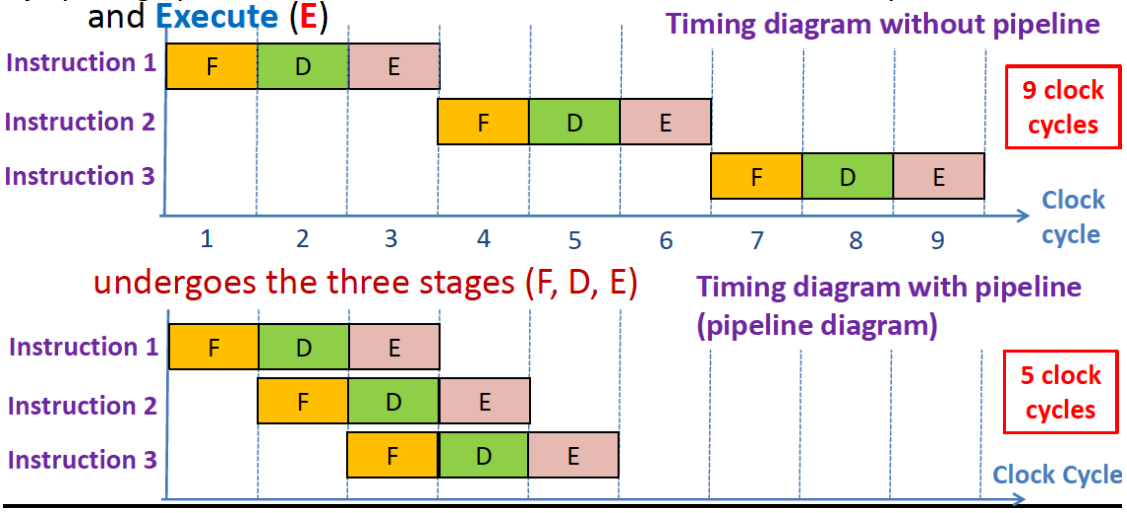
- Provides high precision with large range
- Modes
    -

Mode	Sign	Exponent	Fraction
Normalized	1 / 0	00000001 to 11111110	Anything
Denormalized	1 / 0	00000000	Non zero
Zero	1 / 0	00000000	0000 … 0000
Infinity	1 / 0	11111111	0000 … 0000
Not a Number (NaN)	1 / 0	11111111	Non zero

- Components
    - Sign
    - Exponent
        - Bias = 127 (Single), 1023 (Double)
        - Exponent = 101100102 = 178; E – Bias = 178 – 127 = 51
    - Fraction
        - Assumes hidden 1
        - 0110 = 1.0110
- Single Precision
    - 32 bits
    - 1,8,23
- Double Precision
    - 64 bits
- Different modes to fix underflow
    - Normalized
    - De Normalized
- Conversion
    - Fraction x $2^{\text{Exponent - Bias}}$

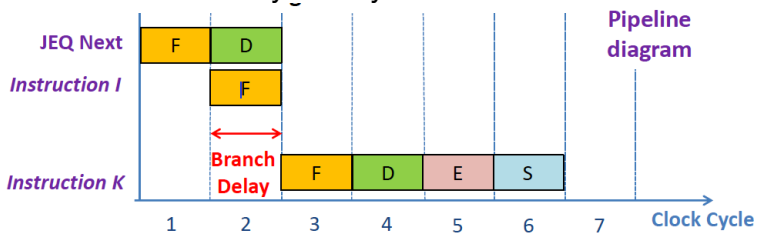
12 - CE1006 - Parallelism

CPU

- Instructions are in 3 stages
  - Fetch
  - Decode
  - Execute
- By splitting up the resource, more instructions can be executed in parallel
- 

Pipelining Conflicts

- Minimum Time
  - No Pipeline Conflicts that slows execution
- Conditions
  - Resource conflict
    - Two instructions requesting same resource at the same time
    - Resolutions
      - Wait
        - Hold instruction until delay is over
        - Requires mechanism to check for conflict
      - Add Resources
        - Add more hardware
  - Data dependencies
    - Instruction depends on previous result
    - Resolution
      - Stall pipeline - Hardware (Circuitry)
      - Insert NOP instructions - Software (Compiler)
  - Branch statements
    - Evaluate condition to know which branch to be taken

- Unnecessary instructions are fetched if branch jumps
- Resolutions
  - Reducing Branch Delay
    - Calculating Branch target at Decode stage
    - Requires extra adder in circuitry
    - Reduces branch delay to 1 cycle
    - 

  - Delayed Branching
    - Fetch the instruction directly before the branch
    - Ensure no branch delay
    - Note: Instruction must be independent and does not affect the branching value
    - Compiler can schedule such instructions
    - If no independant instructions
      - one cycle delay
      - NOP instruction added
  - Dynamic Branch Prediction
    - Hardware circuit to predict outcome
    - Prediction
      - Correct - No wasted cycles
      - Incorrect - Wasted cycles
        - Update prediction bit for future use