Data Representation - CS132

Decimal to Binary conversion

Repeatedly divide by 2 to get remainders $r_0, r_1, \ldots r_n$

then binary representation is $r_n r_{n-1} \ldots r_1 r_0$

Decimal to Octal or Hex

Same as binary but dividing by 8 and 16 respectively.

Addition in binary

100 4 + 011 7

010 2 + 001 1

110 6 (1)110

overflow

Subtraction:

To subtract we add a number to the other numbers negated value

Signed Magnitude

1 1 0 1

negative? 2^2 2^1 2^0

$-1(2^2 + 0 + 1) = -5$

Two's Complement Class range $-2^{N-1} \rightarrow 2^{N-1}-1$

1 1 0 1

-2^3 2^2 2^1 2^0

$-8 + 4 + 0 + 1 = -3$

Addition in two's complement

Same as before, but ignore overflow

Fractional numbers

Fixed Point

$10.11 =$ 1 0 1 1

2^1 2^0 2^{-1} 2^{-2}

$2 + 0 + \frac{1}{2} + \frac{1}{4} = 2\frac{3}{4}$

Floating Point

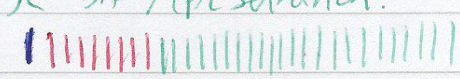
Similar to scientific notation

mantissa $\times 10^{\text{exponent}}$

IEEE Floating Point

More range, less precision.

32 bit representation:



Sign Exponent Mantissa

Digital Logic - CS 132

Moore's Law

predicted the number of transistors on a chip would double every 12-18 months.

Consequences

- Cost of computer logic and memory circuitry has fallen at a dramatic rate
- As logic and memory is more tightly packed, the operating speed is increased.
- Computers become smaller
- reduction in power and cooling requirements.
- interconnections on ICs are more reliable than solder connections, so having fewer off-chip connections increases reliability

Combinatorial logic circuit

A logic circuit whose output is a logical function of its input.

Simplifying boolean expressions with boolean algebra.

distributive law (AND)

$$A + B.C = (A+B).(A+C)$$

POSITION IMPORTANT

C \ AB	00	01	11	10
0	1	0	1	0
1	1	1	1	0

$\bar{A}\cdot\bar{B} + C\cdot\bar{A} + A\cdot B$

Karnaugh maps

A	B	C	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

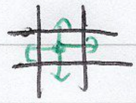
Sum of products

$\bar{A}.\bar{B}.\bar{C} + \bar{A}.\bar{B}.C +$
$A.B.\bar{C} + \bar{A}.B.C +$
$A.B.C$

we can simplify this algebraically, but karnaugh maps are easier.

groupings:

only allow grouping of adjacent squares



Can also loop around the KMap

	00	01	11	10
0	1	0	0	1
1	1	1	1	1

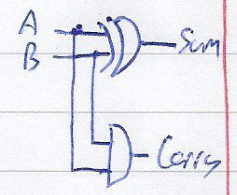
Don't care conditions

Allows us to simplify expressions, we can use it in KMap if it simplifies it.

Digital Logic Ecnid - CS 132

Half-Adder

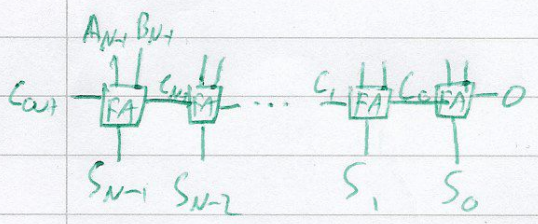
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

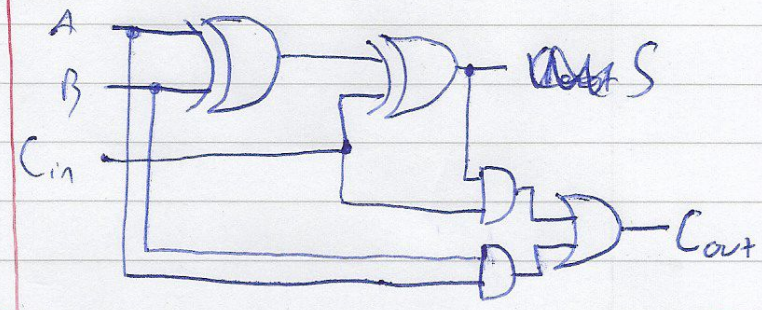


1-bit Full-Adder

C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

N-Bit Full-Adder

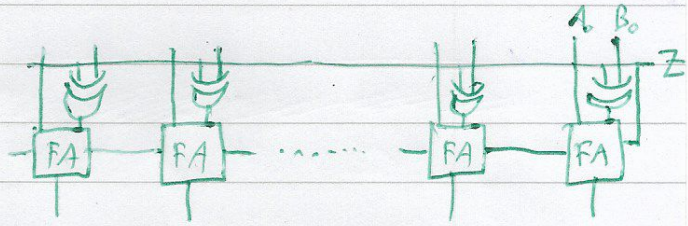




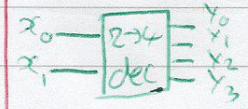
Subtractor:

We add in a control line to determine between adder and subtractor.
Can use the control line to perform both steps to convert B to a negative number.

1. Flip bits
2. +1

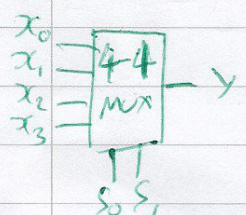


Active-High decoder



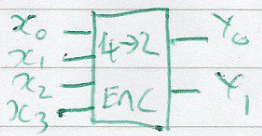
x_0	x_1	Y_0	Y_1	Y_2	Y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Multiplexor



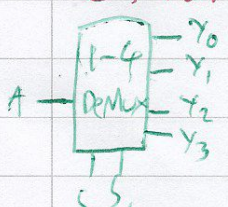
S_0	S_1	Y
0	0	x_0
0	1	x_1
1	0	x_2
1	1	x_3

Active-High encoder



x_0	x_1	x_2	x_3	Y_0	Y_1
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

De-Multiplexor (Activehigh)



S_0	S_1	Y_0	Y_1	Y_2	Y_3
0	0	A	0	0	0
0	1	0	A	0	0
1	0	0	0	A	0
1	1	0	0	0	A

Digital Logic Cont'd - CS132

Common Applications

Multiplexor

- Source selection control
  - Home stereo control e.g. Send iPod, CD or radio to speakers - Analogue, not digital
- Share one communication line between multiple senders
  - Requires both Mux and De-Mux.
- Parallel to serial conversion

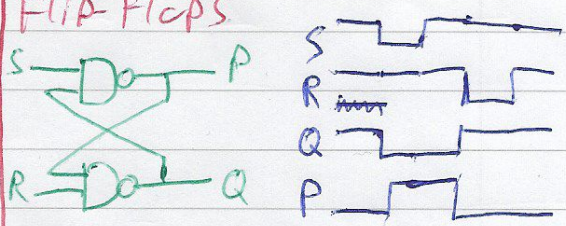
De-Multiplexor

- Share ~~multiple~~ one communication line between multiple senders
- Serial to parallel conversion
- A control for multiple lights

Sequential Logic

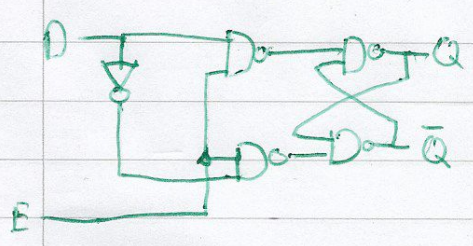
A logic circuit whose outputs are logical functions of its inputs and current state

Flip-Flops



D-Type latch

essentially 1-bit of memory

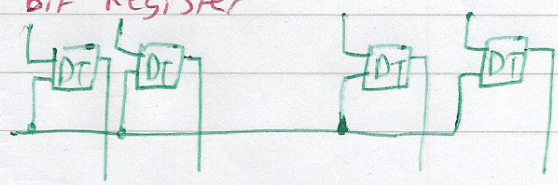


E	D	Q	Q̄
0	0	Q	Q̄
0	1	Q	Q̄
1	0	0	1
1	1	1	0

Clocked Flip-Flops

Changes on the rising edge of a clock input

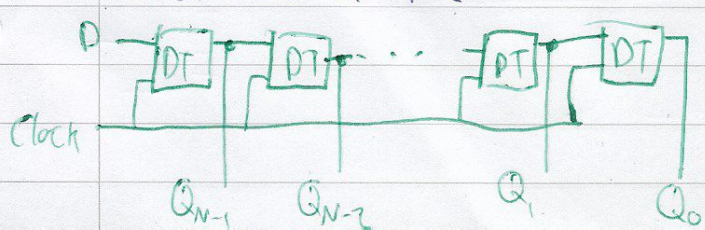
N-bit Register



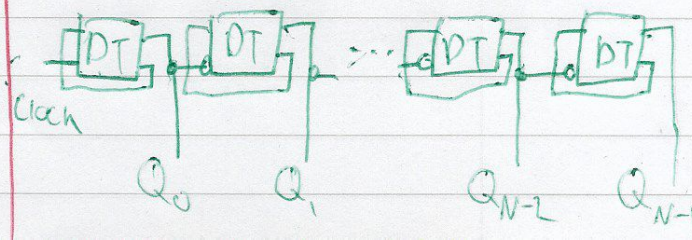
Multi-bit memory

N-bit Shift register

A register that stores and shifts 1 bit at a time



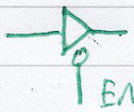
N-bit Counter



Digital Logic Cont'd. - CS132

Three-State logic

instead of just 1 and 0, we now have another state UNCONNECTED.

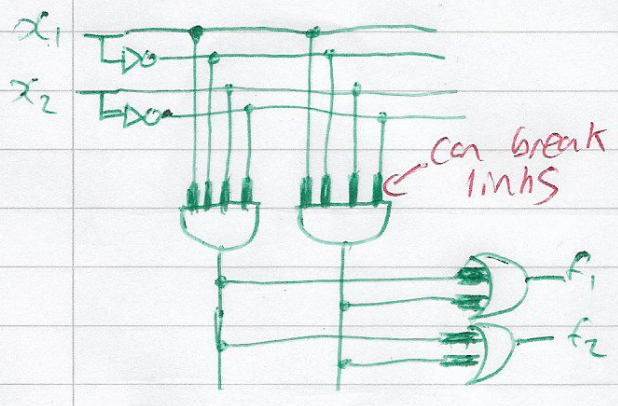


high = connected
low = unconnected

Propagation delay

each logic circuit takes a bit of time to run. Can be reduced by placing components closer together

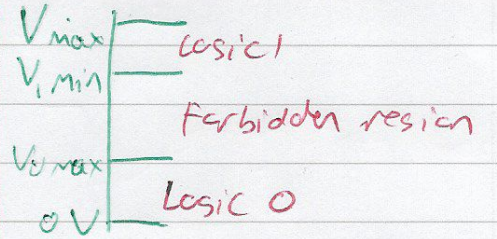
PLA organisation



Three-State Buses

Can use Three-State logic to allow multiple sources of data onto 1 bus.

Properties of Logic Gates



Do not connect outputs together except three-state buffers

ICs

Logic gates and functions can be obtained in small ICs

There are many types of programmable logic device:

- Programmable Array Logic (PAL)
  - one-time programmable
- programmable logic array (PLA)
  - Contains an And array, which feeds to an OR array.
- Field Programmable gate array (FPGA)
  - Modern
  - Can contain millions of gates
  - Enough for an entire processor

Microprocessor Fundamentals

Assembler - CS132

CPU

Controls and performs instructions.

Commonly found on a Silicon Chip called a microprocessor

Fetch-Decode-Execute cycle

Fetch:

- instruction fetched from memory location in PC
- Store in IR
- increment PC

Decode:

- Retrieved instructions/opcode decoded.
- Read effective address to establish opcode type.

Execute

- CU signals CPU components
- May result in changes to data registers, PC, ALU, I/O etc.

Status Register

Consists of various status bits that are set or reset upon certain conditions arising from the ALU

Key components of CPU

ALU - Arithmetic Logic Unit

CU - Control Unit

Fetch-Decode Execute Components

ALU, MBR - Performs Mathematical and logical operations

PC

CU - Decodes programs instructions and handles logistics for executing them.

IR - Contains most recently fetched Instruction.

PC - Tracks the memory address of next instruction.

MAR - Contains memory address location to R/W from/to.

MDR - Contains data fetched from memory, or ready to write to memory

Data registers

Stores frequently used values and intermediate results

Address Registers

- used as POINTER REGISTERS in the calculation of operand addresses
- One of these registers can be used to hold subroutine return addresses.

Register Transfer Language

Assembler - CS132

What is RTL?

describes the operations of a microprocessor as it is executing instructions.

e.g.

[MAR] ← [PC]

RTL - Instruction Fetching

This part of the cycle is always the same

1. [MAR] ← [PC]
2. [PC] ← [PC] + 1
3. [MBR] ← [MS([MAR])] (R/$\overline{W}$ set to read)
4. [IR] ← [MBR]
5. [CU] ← [IR(opcode)]

Fetch and Execute - Adding a constant byte to DO

~~1. [MAR] ← [PC]~~
~~2. [PC] ← [PC] +1~~ Instruction fetching (1-5)
~~3. [MBR] ← [~~
6. [MAR] ← [PC]
7. [PC] ← [PC] + 1
8. [MBR] ← [MS([MAR])]
9. [ALU] ← [MBR] + DO
10. [DO] ← ALU

Assembly Language

Assembler - CS132

Assembly language

uses easily remembered mnemonics for each instruction

Also allows memory locations and constants to be given symbolic names

Why use Assembly language?

Computers only understand 1's and 0's.

But this is hard for humans to understand.

This is why we use assembly languages

Assembler format:

<label>:<opcode><operand(s)> |comment

e.g.

START: move.b #5, D0 | load D0 with 5

Instruction Set

Commonly have 2 aspects:

- Instructions
- Addressing Modes

Subroutines and Stacks

Subroutines

Frequently used sections of code.

Reduces program size

improves readability

JSR <label> Jump to subroutine

RTS Return from subroutine

Use of Stack for Subroutine calls.

1. Subroutine call
   - Pushes contents of PC onto stack
   - puts start address of subroutine to PC
2. Return from subroutine
   - pops return address from stack and puts it in PC

So the stack stores where to return from a subroutine.

A subroutine can call another subroutine, so we have multiple addresses on the stack.

Addressing modes

Assembler - CS132

Some addressing modes
- Direct Addressing
- Immediat Addressing
- Absolute Addressing
- Address Register - indirect
- Relative Addressing

Direct Addressing

The address of an operand is specified by either a data or address register.

e.s. move D3, D2

Immediate Addressing

The operand forms part of the instruction and remains a constant.

e.s. move #42, D5

Absolute addressing

The operand specifies the location in memory explicitly.

e.s. move D2, $7FFF0.

does not allow position independent code.

Address register - Indirect

take the contents of an address register and use that as the address to get Data from

with offset

Similar, but add to the fetched contents a constant

post-incrementing

Normal indirect addressing, but increment the address register afterwards

pre-decrementing

Normal indirect addressing but decrement the address register before.

Indexed Addressing

Add together 2 address registers and a constant to get the address.

Relative addressing

Take the contents of the PC and can add or substract values to it.

Relative addressing is useful for position independent code.

Memory Systems - CS132

Choice of Memory Tech

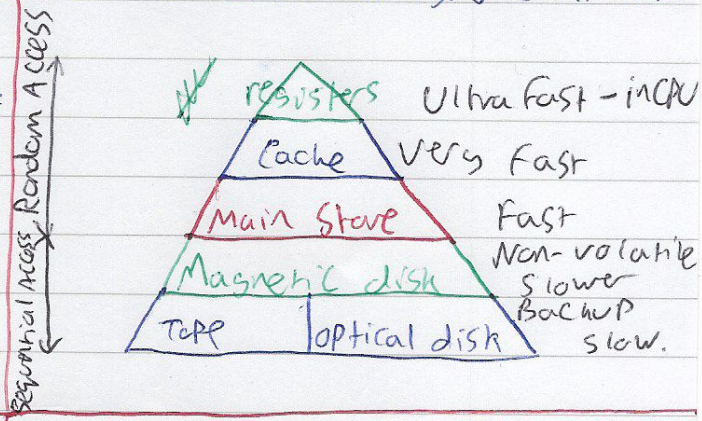
MUST allow for memory to be read and written

Many factors influence choice of memory technology:

- Frequency of access
- Access time
- Capacity required
- Cost

Memory hierarchy

used to solve Designer's dilemma



Temporal Locality:

If a memory location is referenced, it is likely to be referenced again soon

Spatial Locality

If a memory location is referenced it is likely nearby memory will be referenced soon

Reasonable Assumption

90% of memory accesses are within 2kb of previous PC position

Cache

Small bit of memory located near the CPU for faster lookup time.

Cache is small to limit cost.

Moore's law

"Number of transistors on a memory chip doubles every 18 months"

Consequenses:

- Cost of computer logic and memory circuitry has fallen
- Speed increases
- Size decreases
- Power and cooling requirements decreases.
- increases reliability

Who have cache?

Memory Size increasing.

Memory Access Speed not increasing quick enough.

Cache Concepts

Cache read

No Change will occur, So Copies in cache will stay the Same

∴ no change needed.

Cache write

1. Write through - update item in cache and writes through to update lower levels.
2. Only updates copy in cache ensuring blocks are copied to memory when replaced - write Back

Cache Miss

$$h = \frac{\text{Number of times words are in cache}}{\text{Total number of memory references}}$$

Memory Systems 2 - CS 132

Types of cache miss

- Compulsory - Misses that would occur regardless of cache size.
- Capacity - Misses occuring as a result of cache not being large enough.
- Conflict - Misses occurring as a result of the placement strategy for blocks not being fully associative, meaning a block may be discarded and retrieved.
- Coherency - Misses occurring due to cache flushes in multiprocessor systems

Semiconductor memory

Most common form of main store RAM

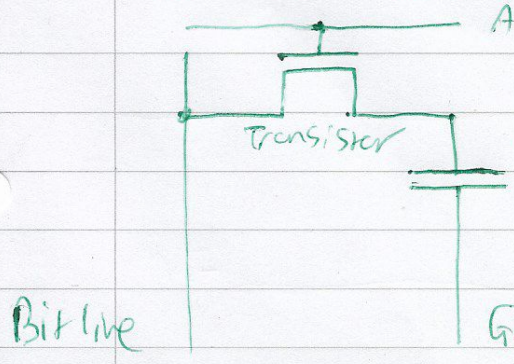
2 main technologies
- SRAM
- DRAM

Dynamic Ram (DRAM)

Store data as charge on a capacitors

Capacitors discharge, hence DRAM cells require periodic charge to maintain data storage.

1 bit DRAM cell



Measuring Cache performance

hit/miss rate isn't enough. we need to know how much slower a miss is.

Average memory access time

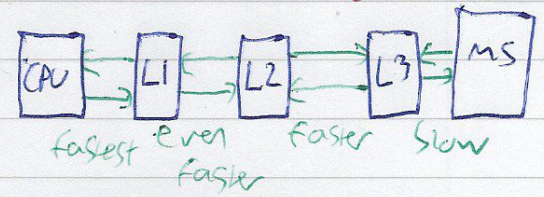
=

Hit time + (miss rate X Miss penalty)

This can still be innacurate

Execution time can often be a better measure

Multi-level cache



Static RAM (SRAM)

- Stores data using configurations of flip-flops and logic gates
- Memory cells hold data as long as power is supplied.
- Made up of 6 transistors

DRAM operations

Write:

Voltage applied to bit line

A signal is applied to the address line to allow the bit line value through to the capacitor

Read:

Transistor powered by address line, allowing the capacitor to discharge into the bit line.

Capacitor needs recharging after.

Memory Systems 3 — CS132

Comparing SRAM & DRAM

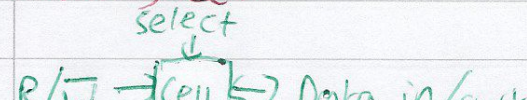
Both volatile

Dynamic are simpler and more compact.

Refresh circuitry of DRAM incurs one-off cost.

SRAM typically faster

Memory Cell



Memory Chip organisation

Can organise memory how we want e.g. 128 x 8 which would require 7 address lines.

Poor choices

Try and make memory organised in a square. don't do something like 1024 x 1.

This results in a long narrow cell array with a large address decoder.

Inefficient use of space

System Memory organisation

Can use similar idea → to organise multiple memory ICs.

See OS - Memory Systems Slide 10. for diagram.

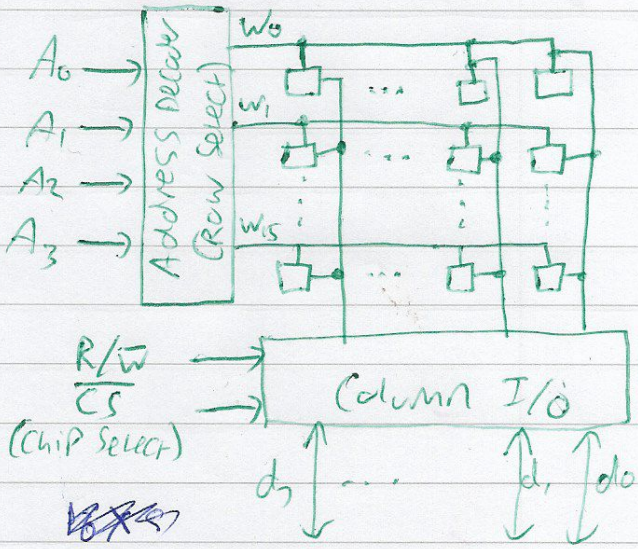
Advanced DRAM Organisations

data transfer between MS and CPU is large bottleneck.

Other DRAM organisations can help with this

- SDRAM - Synchronous DRAM
- RDRAM - Rambus DRAM
- DDR SDRAM
- CDRAM - Cache DRAM

Memory Chip

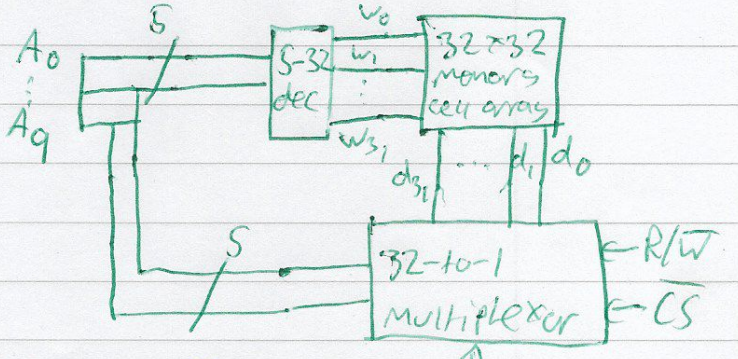


16 x 8 memory IC.

16 words, each of 8 bits.

Minimising Address Decoding Space

Split address input into 2 parts: row address and column address



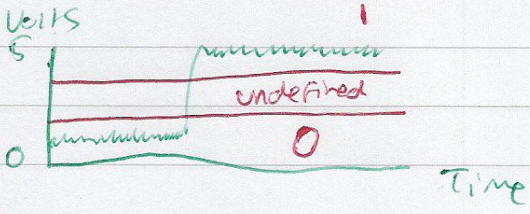
minimises space for address decoding

Memory Systems 4 - CS132

Noise

Noise is unwanted info.
Noise arises from the physical properties of devices
- Thermal noise
- Noise from electronic components
- Noise of transmission circuits

Noise and Disital Logic Devices



Noise Immunity is high in disital devices
Immunity breaks once the noise reaches a certain magnitude

Majority voting.

Send the message 3 times and take a vote from the responses

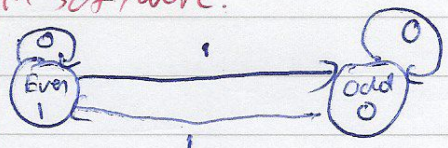
EXPENSIVE

Works well, as, if the prob 1 error occurs is low, then the prob 2 errors occur at the same place is even lower

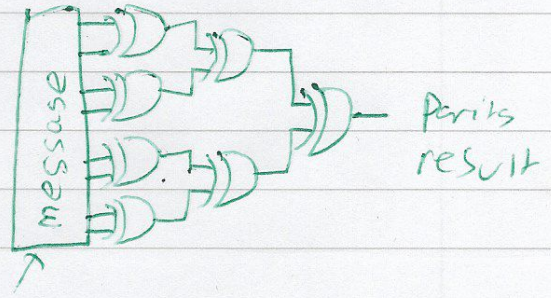
Parity

Adds 1 bit to the message to make the number of 1's in the MSG even or odd.

in Software:



in Hardware



dumms input (in place where parity bit should be)
=0 for even parity
=1 for odd parity

Error Detection Using Parity

At Transmitter
Compute parity bit (P) add to message
At reciever
recompute parity bit (Q)
if P≠Q, then an error has occurred.

Burst Errors

In practice, many errors occur at once in a burst.
This makes parity check useless.
Solution is to calculate a checksum.

Error Correcting Codes (ECCs)

Calculate both Character and Column parity.

M	100	1 1	0 1
e	110	0 1	0 1
s	111	0 0	1 1
s	111	0 0	1 1
a	110	0 0	0 1
g	110	0 1	1 1
e	110	0 0	0 1
↵	100	1 0	1 1

NON-ASSESSED

Memory Systems 5 - CS132

Hard disk

Records data by magnetising a thin film of ferromagnetic material on a disk.

Data Organisation

One track contains many sectors, Each seperated by an inter-connected gap.

Sector contains preamble to synchronise the head before read/write, data and ECC.

performance

Seek times - time to move arm to correct track.

Rotational latency:

motors spin at a constant speed.

Averase delay = 3 to 6 ms

Formatting reduces Capacity by 15%.

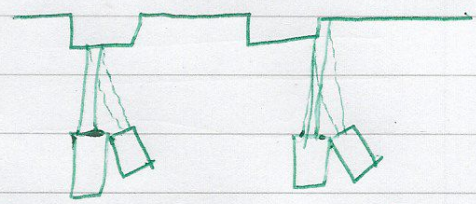
increasing Capacity with zones

The outer edge of the hard disk has a faster velocity

This means the bit density is lower at the outer edge.

To Change this there are more zones placed in the outer edges.

Optical Disk



Strong reflection — weak reflection

0 — 1

Change of State = 1

otherwise = 0

Error prevention

Cross Interleaved Reed-Solomon (CIRC) code is used to spread data out over the disk

up to 2mm of track can be corrected.

Data only accounts for 28% of bits stored, thus we pay a heavy price for great reliability.

performance

Seek time averages are typically 80ms, which is atleast an order of magnitude worse than hard disks.

I/O Mechanisms - CS132

Memory-Mapped I/O

- Same address bus used for memory and I/O devices.
- Memory and registers on I/O devices are mapped to address values.

Disadvantages of Memory-Mapped I/O

portions of memory address space must be reserved.
This problem is only relevant for 16/32 bit machines

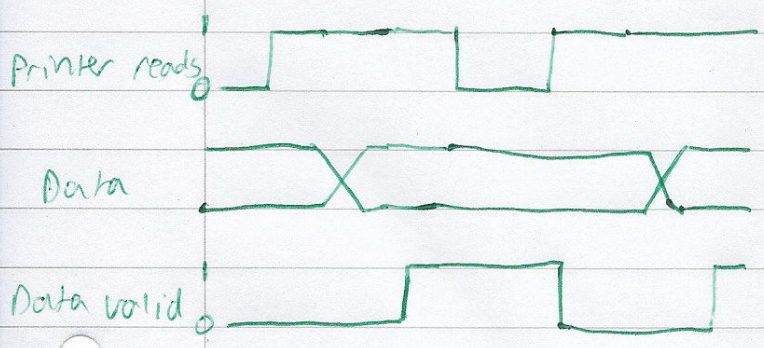
Polling advantages

Simple software and hardware.

- looping construct and checks
- Support for action "reads"

Handshaking

responds to I/O device being ready by placing data on the bus and signalling that the data on the bus is valid.

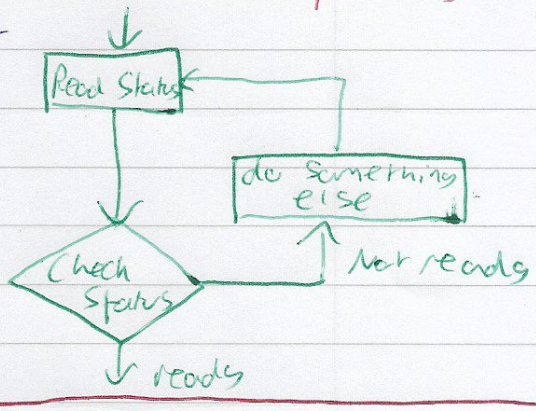


Advantages of Memory-mapped I/O.

Simple
CPU requires less internal logic
Allows use of general purpose memory instructions.
Addressing modes are available to I/O

Synchronising with polling

Busy-wait polling



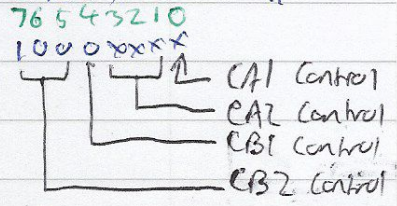
Polling disadvantages

Wastes CPU time and power.
polling with interleaved tasks can lead to delayed responses

6522 VIA performing handshaking.

PCR 1000xxxx
CB2 handshake output mode
CB1 interrupt flag set by negative transition on CB1.
PRINTER READY Asserted.
VIA sets IFR4
VIA sets DATA_VALID to invalid.
When CPU writes a value to ORB
VIA sets DATA_VALID to valid

CB1: printer_ready
CB2: DATA_VALID



I/O Mechanisms 2 - CS132

Interrupts

CPU normally executes instructions sequentially unless there is a jump. An interrupt forces the CPU to jump to a service routine.

Nested Interrupts

Interrupts can be called while processing other interrupts.
This is why we have stacks

Interrupt Advantages

Fast response
No wasted CPU time/power

Interrupt Disadvantages

All data transfers still controlled by CPU
More complex hardware and software.

DMAC

dedicated device that controls the 3 system buffers during data transfer.
DMAC is optimised for ~~case~~ data transfer.

DMA operation

1. DMA transfer requested by I/O
2. DMAC passes request to CPU
3. CPU initialises DMAC
   i. Input or output
   ii. Start address → DMAC Address reg.
   iii. Number of words to transfer → Count reg.
   iv. CPU enables DMAC
4. DMAC requests use of system buses
5. CPU responds with DMA Ack when it's ready to surrender buses

Interrupt Sequence

1. External device signals interrupt

INTERRUPT RESPONSE

1. CPU completes current instruction
2. Push PC onto stack
3. Push status register(s) onto stack
4. Load PC with address of interrupt handler

RETURN FROM INTERRUPT

1. Pop PC from stack
2. Pop status register(s) from stack
3. Load PC with popped return address.

Direct Memory Access (DMA)

CPU is a bottleneck for I/O
DMA avoids the CPU bottleneck

Why use DMA?

Used when large amounts of data must be transferred at high speeds.
Control of buses surrendered to a DMA Controller (DMAC)
DMA data transfer can be more than 10x faster than CPU-driven I/O

I/O Mechanisms 3 - CS132

DMA modes of operation

Cycle Stealing

DMAC uses the system buses when they are not being used by the CPU.

Burst Mode

DMAC requires system buses for extended transfer of large amounts of data at a high speed and "locks" the CPU from using them.
The CPU is locked until the transfer is complete or a device with higher priority requests an interrupt.

DMA organisation - Single bus detached DMA

All modules share the same system buses
The DMA module mimics the processor, uses programmed I/O to exchange data between memory and an I/O module via the DMA module.
Straightforward to implement, but inefficient.
each word transfer takes 2 bus cycles

DMA organisation - I/O Bus

Reduces number of I/O interfaces in the DMA module to one.
System bus the DMA module shares with the processor and memory is used only to transfer data to and from memory.

Processor Architecture - CS 132

Organisation Vs. Architecture

Architecture concerns the structure and properties of a computer system, from a software viewpoint

Organisation is the same but from a hardware viewpoint.

Control Unit

Takes several inputs:

- opcode from IR
- values from CCR
- A stream of clock pulses

The control unit generates signals to control all CPU components

Control Signals Fetch

1 E_{PC}, C_{MAR}

2. R, E_{MS}, C_{IR}

3. E_{PC}

3. $F_1=0$ $F_2=0$

3. C_{ALUR}

4. E_{ALUR}, C_{PC}

Analysing Control Signals

Analysis allows us to determine the number of control steps a particular instruction takes.

Number of steps taken for an instruction is important as more steps means slower execution

Representation of Instructions

Address	Mnemonic	opcode	Operand
0	CLEAR	000	00000
1	ADD #9	010	01001
2	DEC1	011	00000

Fetch Macro Instructions

This operation is performed before every instruction

$[IR] \leftarrow [MS(PC)]$

$[PC] \leftarrow [PC]+1$

Control Signals example - INC1

use PATP model.

1. $[ALU(Q)] \leftarrow [DO]$	1. E_{DO}
2. $[ALU(F)] \leftarrow 01$ ("Q+1")	1. $F_1=0$ $F_2=1$
3. $[ALURes] \leftarrow [ALU]$	1. C_{ALUR}
4. $[DO] \leftarrow [ALURes]$	2. E_{ALUR}, C_{DO}

Control Signal Assumptions

Enables are level trissered

Clock signals are falling edge trissered

An output can be enabled onto the main bus and then clocked in elsewhere in one step.

ALU timings assume that, if values are enabled at P and Q at cycle start, C_{ALUR} can be clocked at cycle end

MS timings assume that, if MAR is loaded during 1 cycle, then R, W and EMS can be used in the next cycle

Processor Architecture 2 - CS132

Control Unit Task

FETCH, DECODE, EXECUTE

Hardwired CU

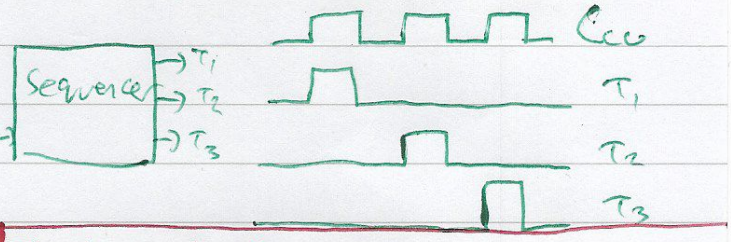
We use a sequencer

We build a digital logic circuit that produces outputs if we connect a clock input

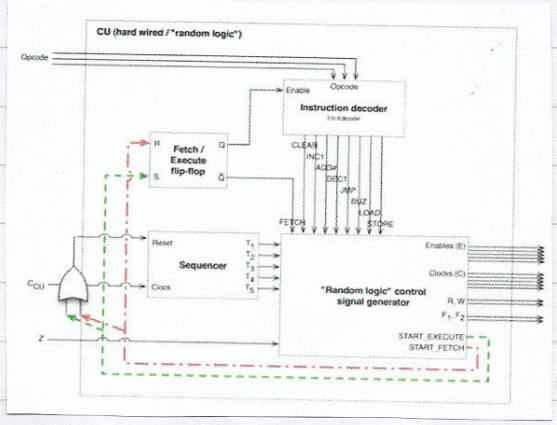
Control Unit Design

2 Approaches

- Hardwired / "Random Logic"
- Microprogrammed



Hardwired CU overview



Hardwired CU Advantages

FAST

Hardwired CU Disadvantages

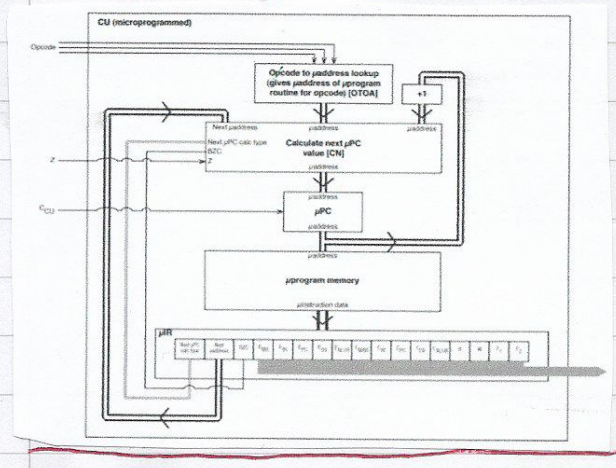
Complex

difficult to design and test.

Inflexible

Long design time.

Microprogrammed CU overview



Microprogrammed CU

Uses microprogram memory

Store the required control action in memory

Microprogrammed CU - Terminology

Microprogram routine

Describes how to generate the CU outputs for a macro instruction.

Microaddress

A location within microprogram memory

Micro PC

CUs internal program counter

Micro IR

CUs internal Instruction Register

Microinstruction

Holds CU output values and other fields to control microprogram flow

Micro PC

What can next μPC value be

- Current μaddress +1
- μaddress given in μIR Next μaddress field.
- Output from OTOA circuit

Decision depends on design choices

Processor Architecture 3 - CS132

Micro Program Execution - Normal operation

- When powered μPC initialises to Maddress 0.
- Execution of the μprogram then starts.
- Each Minstruction sets the CU outputs to the values recorded within the Minstruction. This generates PATP control signals corresponding to the PATP fetch operation.
- After each Minstruction, μPC is incremented

Microprogram Execution - End of Fetch

- Last Minstruction is unusual.
- IR is filled with next opcode and this is fed to the CU opcode inputs
- Last fetch Minstruction has the next μPC calc type field set so we don't just increment PC. Instead we set the output of the OTOA circuit.
- μPC jumps to the Maddress of the μProgram for the opcode fetched from memory.
- CU starts to execute that μprogram

Microprogram Execution - End of Microprograms

- is unusual
- Next μPC calc type field set s.t. μPC is set from the MIR next Maddress field.
- Opcode μprograms use this facility to set the μPC back to the fetch μprogram address

Microprogram Execution - Conditional Branch

- PATP use BZC
- Next μPC value is determined by the MIR's next Maddress if Z is set. Otherwise +1 circuit used.

Microprogram Advantages

- Ease of design and implementation
- Flexible
- Simple hardware
- Mprogram memory can be reprogrammed

Microprogram Disadvantages

- Slower than hardware implementations.