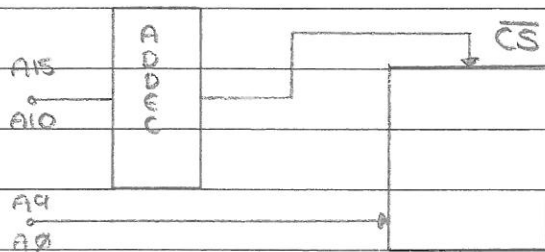


Decoding for longer blocks of locations

Say we had a 1K memory, it would have 10 address line (to select from 1000) globally enabled by a CS signal (chip select).

On a 16-bit address Bus, we would normally route A₉-A₀ to the memory chip and A₁₅-A₁₀ to the address decoder (ADDEC), to produce chip select.



Such a memory allocation places the memory on a "natural" 1K boundary. A natural boundary is a boundary sub division of the overall address space for ex $\frac{1}{2}$ $\frac{1}{4}$ $\frac{1}{8}$. These are 16 $\times$ 1K natural boundaries in a 64K space starting at address 0000, 1000, 2000, 3000... f000.

A natural block is characterised by being able to split the address lines between the address decoder (ADDEC) and the device.

Examples of memory or block decoding

Lets use throughout a 4k device (or block). We have seen already that there are $16 \times 4k$ "natural" boundaries in a 64k address space. Thus our 4k device could occupy any of 000-0fff, 1000-1fff, ..., f000-ffff.

The top 6 address bits go to the address decoder (ADDEC) to identify the block, the other 12 bits go to the device, to identify internally the location.

Example

Test if a 4k device spans a natural boundary
1800 - 2fff

Test 1

check address range is same as device size ie 4k

$$\begin{aligned} & 2800 && (2800 = 2fff + 1) \\ & - 1800 \\ & 1000 = 4k \end{aligned}$$

Test 2

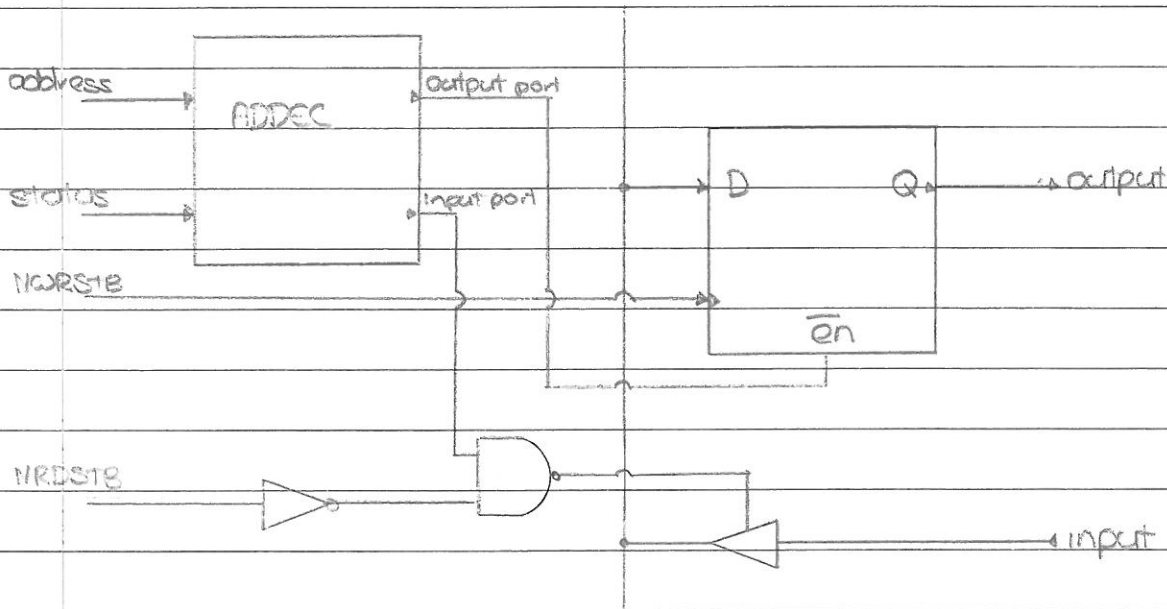
Considering the top 8 address bits (pages). The 4k device is therefore placed as $2 \times 2k$ natural blocks. note that all now goes to the address decoder and device.

Test 3

Ensure common address bits between the address decoder and the device, take all possible values.

Worked example

Draw a circuit diagram, using basic components, for an input port and an output port attached to a synchronous bus. Briefly explain the operation of the main components of your circuit?



The input & output port or I/O for short consist of three main components, they are the address decoder (ADDEC), output port, and the input port.

The address decoder is responsible for decoding the address and status bus information and selecting the individual locations. It is basically an AND function.

The output part is basically a clock enabled D-type.

The input port is a tri-state buffer and connects the input to the Bus, if its enabled.

Worked example

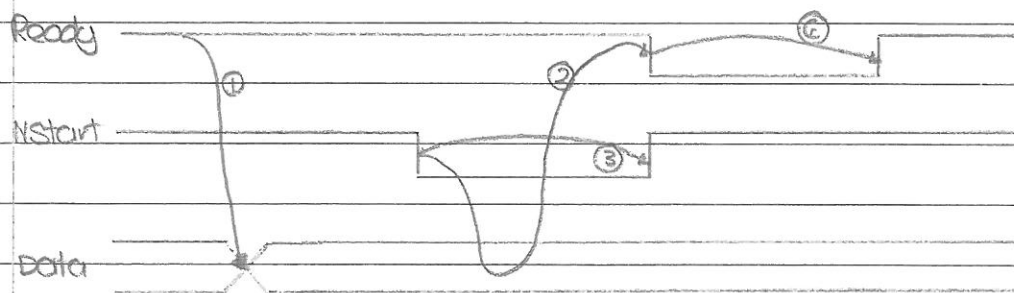
Describe fully the fetch-execute cycle, used to implement microcontroller instructions?

Every microcontroller instruction must do a fetch execute cycle. To start the cycle the instruction register is loaded with an op-code.

The program counter contents is transfer to the address bus, the data is returned by the first bus access cycle and loaded into the instruction register. Each op-code is unique and defines what the instruction does. it is then decoded by the instruction decoder and tells the control section how to implement the instruction.

Worked Example

With the aid of a diagram, explain the operation of the Ready-Start handshake protocol. Give a 68HC11 assembly language routine (as used in lectures) to transfer a byte of data from a peripheral to a microcontroller, via a port-linked interface, using a ready-start handshake.



① Source checks destination

if ready to receive data, it send data
takes "start active"

② when "start" is active it takes data and acknowledges
whilst processing data

③ Source takes "start" as inactive

needs a starting pulse of $> 5\text{ms}$

④ destination becomes ready when data been used

loop

```
BRCLR (status-x), x
```

```
STAB data
```

```
BCLR (control-x), x
```

```
<delay if needed>
```

```
BSET (control-x), x
```

```
RST
```

Worked example

On a typical microcontroller Bus with a 2MHz Bus cycle rate, the address is generated 85ns after the start of the Bus cycle and NRDSTB rises 25ns before the end of the cycle.

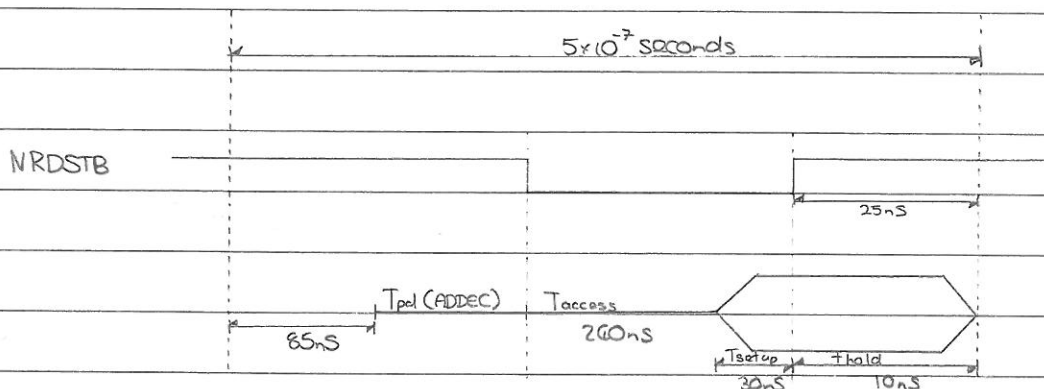
At the CPU, $t_{\text{setup}} = 30\text{ns}$ and $t_{\text{hold}} = 10\text{ns}$. A peripheral with an access time of 200ns is interfaced to the Bus. How fast must the interface address decoder circuit work? Justify your answer using a sketch to illustrate the various timings.

Using

$$T_{\text{period}} = \frac{1}{f}$$

$$T_{\text{period}} = \frac{1}{2 \times 10^6}$$

$$T_{\text{period}} = 5 \times 10^{-7}$$



$$T_{\text{propagation delay}} = 5 \times 10^{-7} - (85 + 200 + 30 + 25) \times 10^{-9}$$

$$T_{\text{propagation delay}} = 5 \times 10^{-7} - 3.8 \times 10^{-7}$$

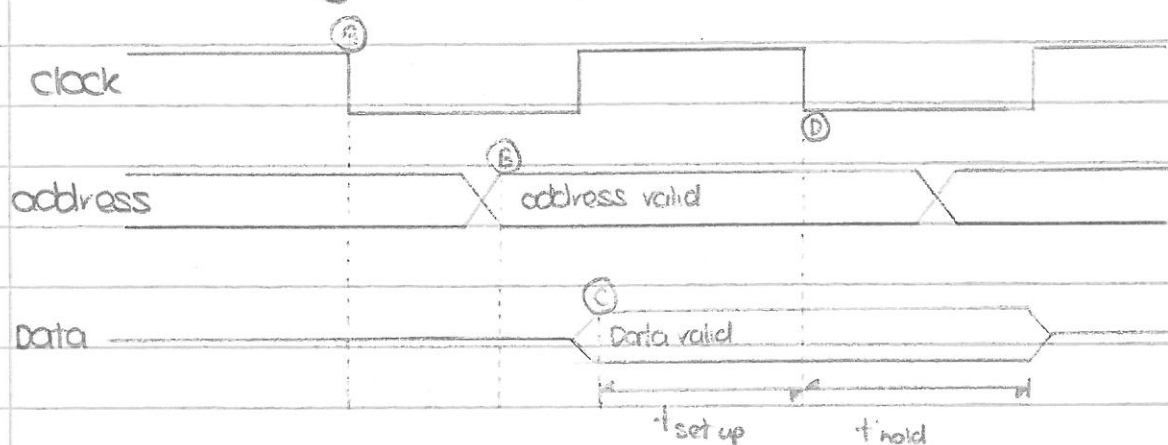
$$T_{\text{propagation delay}} = 1.2 \times 10^{-7} \text{ seconds}$$

$$T_{\text{propagation delay}} = 120 \text{ ns}$$

Question

Describe synchronous data transfer

In a synchronous data transfer, the processor provides an address and timing signals. A simple synchronous data transfer starts with the CPU reading from memory. At point A, a read cycle begins with the falling edge of the clock. At B, the CPU generates an address corresponding to the memory location being accessed. At C, the memory yields its data for the CPU to read. The cycle ends at D with the falling edge of the clock. The clock must allow enough time for the memory to access the data, otherwise the data may be corrupt.



3. Describe addressing modes

Operation codes for a microcontroller can be classified according to the way they address memory.

Extended mode

The operation code is followed by two bytes specifying a full 16-bit memory address.

Direct mode

The operation code is followed by one byte specifying a memory address. This means only addresses 00 to FF can be used by these two byte instructions.

Indexed mode

In this code the operation code is not followed by the address of the data to be operated on. Instead the address of the data is derived from the contents of the 16 bit index register.

Inherent mode

These are single byte instructions which do not use memory at all, for example INC A, DEX, ABA.

Immediate mode

These instructions do not use memory. Instead the operation code is followed directly by the byte or bytes to be operated on, rather than by an address.

Relative mode

These are used only by the branch operation codes. They are always two byte instructions.

5. Describe subroutines?

Subroutines usually contain a sequence of instructions which are often used in a program. Rather than put such a sequence in a program wherever its needed, it is entered just once at a known starting address, usually at the end of the main program. A subroutine must always end with the single byte 'return to subroutine' (rts) instruction, this makes sure that after execution of the subroutine sequence, the microcontroller always returns to the next instruction to be executed in the main program.

5. Describe a stack

stacks are last in, first out memories, we can only add items to the top or remove them from the top. adding an item to a stack is a PUSH operation, and removing an item is a PULL operation.

One major advantage of a stack is that we can add data to it without disturbing the data that is already there. a disadvantage of a stack is that since stacks do not have a fixed address, its location and contents may be difficult to remember or determine.

Worked example

What signals would you expect to find on a typical synchronous microcontroller bus? Describe briefly what each is used for?

A synchronous bus is a path where digital information is transmitted between the modules of a microcontroller. On a typical synchronous bus, you would expect to find.....

- * An Address bus, is used by the CPU to select a particular memory location as a source or destination of data.
- * A Data bus, is used to send data to memory or the send the contents of memory to the CPU.
- * A Control bus, transfers control and timing signals. These signals sent to the memory may include synchronization signals that make the memory access to coincide with the processor cycle and address and data strobes.
- * A status bus, is used to qualify address and is Bidirectional.

Worked example

Write short notes describing the following terms, as applied to microcontrollers.

- ① opcode
- ② accumulator
- ③ program counter

- a. The operation code or op-code for short is part of an instruction that specifies the operation to be performed during the next cycle.
- b. The accumulator is a register that is a source of one operand and the destination of the result for most arithmetic and logical operations.
- c. The program counter is a register that specifies the address of the next instruction to be fetched from program memory.

Worked example

explain how subroutines are implemented on microcontrollers. Show how they may be "nested." What rule must be observed within the subroutine, to ensure its correct operation?

A subroutine is used in a microcontroller program when a function is used more than once in the program. The main reason for the existence of a subroutine is to save program steps and to shorten to program execution time. Subroutines are accessed by BSR instruction and exited by the RST instruction. When a BSR instruction is called the value of the program counter is placed on the top of a stack, and when the rst instruction is called, it simply takes the top of the stack into the program. By using this method subroutines can be "nested" together and another subroutine can be called from a subroutine.

A microcontroller program should never branch out of a subroutine as this disturbs the order of the stack. The subroutine should be use complementary with respect to the stack.

Worked example

Give a tabular description of the address decoder functions required to enable the following devices....

Ⓐ A 2k read-write memory, located between addresses 5E00 - 65ff?

Ⓑ An I/O mapped location, to be written at address 93 and read from address DA?

		A7	A6	A5	A4	A3	A2	A1	A0	RDNWR	I/O/M	CE
a.	read 5E ₁₆	0	1	0	1	1	1	1	0	1	0	q
	write 65 ₁₆	0	1	1	0	0	1	0	1	0	1	q
b.	read DA ₁₆	1	1	0	1	1	0	1	0	1	1	q
	write 93	1	0	0	1	0	0	1	1	0	1	q

Worked example

Derive the machine-code for the following "HCL" subroutine. Const and dac have previously been defined, using the EDCI command, as \$2C and \$103C respectively?

```
loop  LDAA 3,x
      ADDA #const
      STAA dac
      INX
      BNE loop
      RTS
```

Program	Opcode			
LDAA (indirect)	A6	03		
ADDA (immediate)	8B	2C		
STAA (extended)	B7	103C		
INX	08			
BNE	26	-16	(-10)	(-10 due to PC is) (already at 39 & counts) back
RTS	39			

-16

Worked example

The "HC11" instruction SUBA # \$69 is executed, with ACCA holding \$D5. What are the resultant values of ACCA and the N, Z, V and C flags? Show your evaluation, to explain your answers briefly.

ACCA = \$D5

$(213)_{10}$

SUBA = # \$69

$(105)_{10}$

ACCA = 11010101

SUBA = 01101001

ACCA = 01101100

$(108)_{10}$

ACCA = 6C

Result placed into acca

N (negative flag) = 0

Z (zero flag) = 0

V (overflow flag) = 1

C (carry/borrow flag) = 0

1. Describe the main parts of a microcontroller and how the bus operates

The microcontroller consists of thousands of electronic switching circuits. Instructions are fed into a microcontroller are in Binary digits (called BITS). This is normally an eight bit instruction for example 00101101, this eight bit instruction is called a byte. When a Byte is fed into a microcontroller either by a sequence of pulses, which is an asynchronous, or all pulses at once, which is known as synchronous.

A microcontroller requires a memory in which input information can be stored. The memory requires an addressing system so that it is known where the information is stored and where it can be called from.

The input information come in two different forms, either as an instruction or as data, both are expressed in binary and microcontroller must be able to distinguish between them.

Part of the memory is known as ROM (Read only memory) this has a set of instructions manufactured into it and they cannot be changed. So when a microcontroller is given the instructions to place a byte of data into store, the instruction code causes a number of instructions retained in the ROM to be recalled and the data is placed into the next available store.

Another part of the memory is one where information can be temporarily stored, this part is called RAM (Random access memory). RAM is made up of a large number of stores, each of which has an address. Unlike ROM, the information stored in RAM is volatile and lost when power is removed.

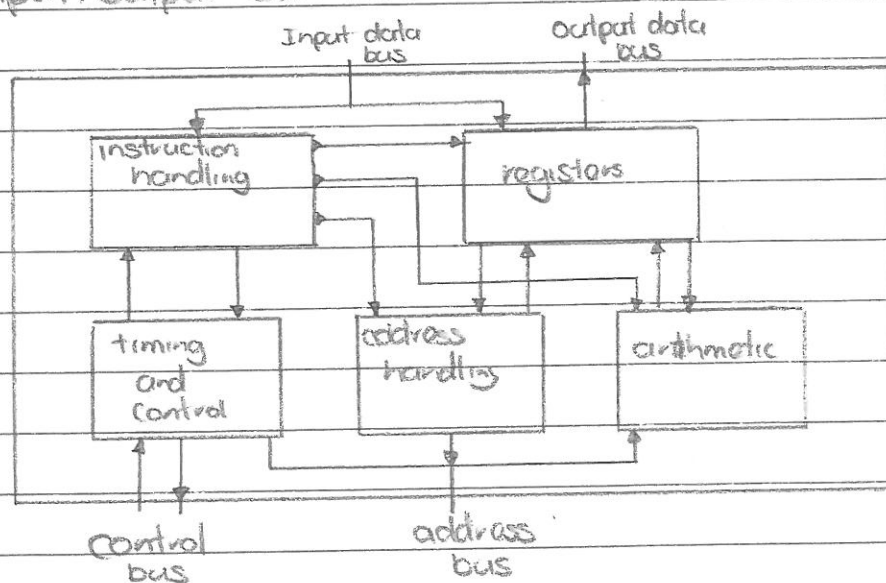
A microcontrol does a lot of interactions, and at the heart of these interactions is an accumulator.

Information can be changed by a process in the accumulator and when the sequence of the process is finished the accumulator is cleared for the next set of operations

2. Describe the internal structure of the CPU and what each part is used for

The CPU (Central processing unit) processes the data. It fetches the instructions from memory, decodes them, and executes them. It generates the timing and control signals, transfers data to or from the memory and input/output sections, performs arithmetic and logical functions, and recognizes external signals. During each instruction cycle the CPU performs tasks such as.....

- * Places the address of the instruction on the memory address bus
- * Takes the instruction from the input data bus and decodes it.
- * Fetches the addresses and data required by the instruction
- * Performs the operation specified by the instruction code.
- * Looks for control signals, such as interrupts and responds.
- * Provides status, control and timing signals that the memory and input/output can use



Worked example

Say what you understand by the term

"semi-synchronous bus." A peripheral requiring a handshake protocol to drive it is to be bus-linked to a semi-synchronous bus. discuss the advantages and the restrictions of such an implementation

Worked example

Say what you understand by the term "semi-synchronous microcontroller bus". How does it differ from a synchronous bus?

Worked example

Derive the machine code for the following "HCII" program segment? where "stat" and "data" have previously been defined, using the ECU command as 5 and \$1007 respectively?

brclr stat, x 2 alt

ldaa #10

adda data

bra join

alt ldaa #\$20

oraa data

Worked example

Explain how interrupts work in a microcontroller.

Distinguish clearly between how the microcontroller hardware responds to the interrupt and what features must be programmed into the interrupt service routine. What is meant by the term "vectored interrupt"?

Worked example

Describe the key features of tri-state and open-collector buffers and clock enabled D-type latches. How are these devices used in microcontroller peripheral interfaces and why are the features that you noted important in such interfaces?

Worked example

With the aid of a diagram, explain how a signal which is short in duration or occurs frequently might not be seen by a microcontroller polling an input port?

Worked example

Explain clearly the difference between "memory mapping" and "I/O mapping". What are the advantages of each?

microprocessor instructions

ADD

(add memory contents to the accumulator)

This instruction adds the contents of a stated memory location to the contents of the accumulator, and stores the result in the accumulator. If several successive additions are carried out, no account is taken of any carries that are generated.

AND

(logical AND memory with accumulator)

This instruction performs a logical AND between the contents of a stated memory location and the contents of the accumulator. The AND is between the equivalent bits of the two bytes, the result is stored in the accumulator. If there are no bit pairs which are both logic 1, the result will be zero and the zero flag will be set to logic 1.

BRK

(break)

This instruction stops the program. No operand is required.

BNZ

(branch if not equal to zero)

When this instruction is encountered, the zero flag is tested. Thus the BNZ instruction should follow an instruction which is designed to affect the zero flag. If the flag is set to logic 0, the branch is taken, otherwise the program continues to the next instruction in sequence.

BZE

(branch if equal to zero)

When this instruction is encountered, the zero flag is tested. Thus the BZE instruction should follow an instruction which is designed to affect the zero flag. If the flag is set to logic 1, then the branch is taken, otherwise the program continues to the next instruction in sequence.

CMP

(Compare memory with accumulator)

This instruction compares the contents of a stored memory location with the contents of the accumulator. This is done by subtracting the memory contents from the accumulator contents. The answer to the subtraction is not stored so the data in the accumulator is not destroyed. The zero flag will be set to logic 1 if the answer is zero, it will be set to logic 0. The CMP instruction will normally be followed by a conditional branch.

DEC

(decrement memory)

This instruction decrements the contents of a stated memory location by 1. If the contents of the memory become zero as a result of this operation, the zero flag will be set to logic 1.

JMP

This instruction causes the program to jump to a stated memory location, and hence instruction, rather than proceed to the next instruction in sequence. The jump is unconditional and is always taken.

LD

(load instruction)

This instruction has two modes.....

① to load the accumulator with fixed data, the data value being held in memory in the location immediately following the instruction.

② to load the accumulator with data contained in a stated memory location. The instruction is followed by the address in memory from which data is to be loaded.

Abbreviations - microcontrollers

CPU	Central Processing Unit
ROM	Read Only Memory
RAM	Random Access memory
I/O	Input / output
RDNWR	Read Not Write
NRDSTB	Not Read STroBe
NWRSTB	Not Write STroBe
ALU	Arithmetic Logic Unit
MSB	Most Significant Bit
SP	Stack Pointer
ADDEC	Address Decoder
DTAC	Digital To Analogue Converter

Glossary

Access time

The access time is the delay between the time when a memory receives an address and the time when the data from that address is available at the outputs.

Accumulator

The accumulator is a register that is the source of one operand and the destination of the result for most arithmetic and logical operations.

Address

An address is an identification code that distinguishes one memory location or I/O port from another and that can be used to select a specific one.

Addressing modes

The methods for specifying the addresses to be used in an instruction. Common addressing modes include direct, indirect, indexed, relative and stack.

ALU (Arithmetic Logic Unit)

The ALU is a device that can perform any of a variety of arithmetic or logical functions under the control of function inputs.

Arithmetic shift

A shift operation that preserves the value of the sign bit (most significant bit)

Assembler

An assembler is a computer program that converts assembly language in which the programmer can use mnemonic instruction codes, labels and names to refer directly to their binary equivalents. The assembler is a low-level language, since each assembly language instruction translates directly into a specific machine language instruction.

Asynchronous

Asynchronous means operating without reference to an overall timing source, that is, operating at irregular intervals.

Bus

A Bus is a group of parallel lines that connect two or more devices.

Byte

The basic grouping of bits that the computer handles as a unit, most often eight bits in length.

Clock

The clock is a regular timing signal that governs transitions in a system.

CMOS

CMOS is an abbreviation of a Complementary Metal Oxide Semiconductor, a logic family that uses complementary N-channel and P-channel MOS field effect transistors to provide high noise immunity and low power consumption.

Delay time

The delay time is the amount of time between the clocking signal and the actual appearance of output data, or the time between input and output.

DOS

DOS is an abbreviation of Disk Operating System, and is an operating system that transfers program and data to and from a disk, which may be either flexible or fixed head, the operating system may itself be largely resident on the disk.

Hold time

The hold time is the amount of time after the end of an activity signal during which some other signal must be stable to ensure the achievement of the correct final state.

Index Register

The index register is a register that can be used to modify memory addresses

I/O section

This section of the computer handles the communications with external devices

Instruction

An instruction is a group of bits that defines a computer operation and is part of the instruction set.

Instruction cycle

The instruction cycle is the process of fetching, decoding and executing an instruction.

Instruction set

This is a set of general purpose instructions available with a given computer, that is, the set of inputs to which the CPU will produce a known response during the instruction fetch cycle.

Interrupt

An interrupt is a computer input that temporarily suspends the normal sequence of operations and transfers control to a special routine.

Jump instruction

This instruction that places a new value into the program counter, thus departing from the normal one-step incrementing. Jump instructions may be conditional, that is, the new value may only be placed in the program counter if certain conditions are met.

memory mapped I/O

This is an addressing method for I/O ports that use the same addressing system as that used by the memory section.

Mnemonics

Symbolic names or abbreviations for instructions, registers, memory locations, which suggest their actual functions or purposes.

Multiplexer

A multiplexer is a device that selects one of several possible inputs to be placed on a time shared output bus according to the state of the selected inputs.

Nesting

The term "nesting" means the construction of subroutines or interrupt service routines so that one transfers control to another and so on. The nesting level is the number of transfers required to reach a particular routine without returning.

Operating System

System software that controls the overall operation of a computer system and performs such tasks as memory allocation, input and output distribution, interrupt processing, and job scheduling.

Operation Code

The op-code is the part of an instruction that specifies the operation to be performed during the next cycle.

Overflow bit

This is a status bit that is one if the last operation produced a two's complement overflow.

Pointer

A Pointer is a register or memory location that contains an address rather than data.

Polling

Polling determines the state of peripherals or other devices by examining each one in succession.

Pop

To remove an operand from a stack

Program counter

The program counter is a register that specifies the address of the next instruction to be fetched from program memory

Schmitt trigger

A Schmitt trigger is a circuit which is used to produce a single, sharp transition from a slowly changing input

Set-up time

The set-up time is the time, prior to a clock transition, during which data must be stable for proper operation.

Shift register

The shift register is a clocked device that moves its contents one bit to the left or right during each clock cycle.

Sign-magnitude number

This is a number in which the most significant bit represents the sign or polarity and the remaining bits represent the magnitude.

Stack

A stack is a sequence of registers or memory locations that are used in a last in, first out manner.

Stack pointer

The stack pointer is a register or memory location that is used to address a stack.

Two's Complement

A binary number that, when added to the original number in a binary adder, produces a zero result. The two's complement is the one's complement plus one.

Two's Complement Overflow

This is a situation in which a signed arithmetic operation produces a result that cannot be represented correctly, that is, the magnitude overflows into the sign bit.