3. Embedded Systems

3.0 Introduction

Microcontrollers are a mono chip a miniature ie small mini computers which contains almost all the basic parts of a personalized computer that too in a very compact environment. Microcontrollers are often referred to as single chip devices or single chip computers. The main and important feature of the comparatively lesser size of the microcontroller is due to availability of less resources when compared with a personal laptop or individual computer. In functional terms, a microcontroller is a programmable single chip which controls a process or system. Microcontrollers are typically used as embedded controllers where they control part of a larger system such as an appliance, automobile, scientific instrument or a computer peripheral. Microcontrollers are designed to be low cost solutions; therefore using them can drastically reduce part and design costs for a project. Physically, a microcontroller is an integrated circuit with pins along each side. The pins presented by a microcontroller are used for power, ground, oscillator, I/O ports, interrupt request signals, reset and control. In contrast, the pins exposed by a microprocessor are most often memory bus signals (rather than I/O ports). Personal computers are used as development platforms for microcontroller projects. Development computers, usually personal or workstation computers, use a microprocessor as their principle computing engine. Microprocessors depend upon a variety of subsidiary chips and devices to provide the resources not available on the microprocessor. Additional chips required with microprocessor support memory storage, input/output control and specialized processing. A development platform is required to run embedded system development software such as assemblers, compilers, editors and simulators which require the processing power and memory capabilities of a desktop personal computer or workstation.
The target platform is the platform on which the finished program will be run. For example, consider a developer who is creating a program for a Motorola 68HC705C8 microcontroller. The developer writes, edits, and tests the program on a Pentium 133 personal computer: the development platform. The developer will use software which runs on a Pentium 133 but whose target device is the 68HC705C8. When the program is ready it is programmed in the target platform, the 68HC705C8.

A microcontroller has seven main components:

1. Central processing unit (CPU)
2. ROM
3. RAM
4. Input and Output
5. Timer
6. Interrupt circuitry
7. Buses

3.1 Architecture:

There are two basic types of architecture: Harvard and Von Neumann. Microcontrollers most often use a Harvard or a modified Harvard-based architecture.

Von Neumann

In Von Neumann architecture we have a mono large memory area so as to keep both the program instructions and the data that need to be stored. Using the same bus both the instructions and data are communicated. Each time the CPU fetches a program instruction it may have to perform one or more read/write operations to data memory space. It must wait until
these subsequent operations are complete before it can fetch and decode the next program instruction. The advantage to this architecture lies in its simplicity and economy.

![Von Neumann Memory Map](image1)

**fig 3.1(a)**

Von Neumann Memory Map Every microcontroller has a very specific layout for its memory. Usually this is depicted with the help of a memory map.

A memory map

memory is used.
Harvard

The program instructions and data in the case of Harvard architecture computers have separate memory areas. With the help of the more number of internal data buses, it is possible to access the instructions and the data simultaneously. Through the program memory, but the central processing unit fetches instructions. If the fetched instruction requires an operation on data memory, the CPU can fetch the next program instruction while it uses the data bus for its data operation. This speeds up execution time at the cost of more hardware complexity. Since Harvard machines assume that only instructions are stored in program memory space, how do you write and access data stored in program memory space? As an instance in ROM only a constant value need to be stored if the data value in C is declared as a constant. Different microcontrollers have different solutions to this problem. A good C compiler automatically generates the code to suit the target hardware’s requirements. There are special instructions for certain chips that allow the information retrieval from program memory space [110]. When compared to the instructions used for fetching the data from data memory, the above-said instructions are always very complex in nature and expensive too. When compared to the instructions that are being used to fetch data from the data memory, the above instructions are relatively more complex and expensive in nature. Typically these chips have a register analogous to the program counter (PC)
that refers to addresses in program space. Also, some chips support the use of any 16 bit value contained in data space as a pointer into the program address space. These chips have special instructions to use these data pointers. [96]

Fig 3.1( c )

3.2 Central Processing Unit

One part of the CPU is responsible for performing calculations and executing instructions. This part is called the arithmetic logic unit, or ALU. There are a variety of subsidiary components which support the ALU. These components include the decoder, the sequencer and a variety of registers. The decoder converts instructions stored in program memory into codes which the ALU can understand.[56] The sequencer manages the flow of data along the microcontroller’s data bus. Registers are used by the CPU to temporarily store vital data which are volatile: they can change during program execution. Most microcontroller registers are memory-mapped,
associated with a memory location, and can be used like any other memory location. Registers store the state of the CPU. If the contents of microcontroller memory and the contents of these registers are saved it is possible to suspend program operation for an indefinite period of time and restart exactly in the state when the program was suspended. Different microcontrollers carry different number of registers associated with them and with different names. Moreover there are a few registers that are most common to majority of microcontrollers even though their names may differ. These include:

- **The stack pointer**

  The stack pointer keeps the stack’s next location address. Whenever in a stack a data is pushed then the address in the stack pointer is decremented and an increment happens when ever a data is popped out of the stack.

- **The index register**

  The index register is used to specify an address when certain addressing modes are used. It is also known as the pointer register. The Microchip devices use the name FSR (file select register).

- **The program counter**

  Perhaps the single most important CPU register is the program counter (PC). The Program counter keeps the new instruction’s address in the memory storage space. It contains the new instruction’s address that the CPU will be processing. As each instruction is fetched and processed by the ALU, the CPU increments the PC and thereby steps through the program stored in the program memory space.

- **The accumulator**
The accumulator is a register that can hold operands or results of operations as necessary. The Microchip devices use the name W (working) register. Other registers may reflect results from the instruction just executed, control the options available on the device, and enable access to certain areas of memory.

**ROM**

ROM stands for read only memory also called as non volatile memory that is used for program information and data that need to be stored on permanent basis. The microcontroller uses ROM memory space to store program instructions it will execute when it is started or reset. Program instructions must be saved in non-volatile memory so that they are not affected by loss of power. The microcontroller usually cannot write data to program memory space.

**RAM**

RAM stands for random access memory or read and write memory is a type where one can read and write any number of times and is basically volatile in nature that is if you remove the power supply its contents are lost. Any variables used in a program are allocated from RAM. The time to retrieve information from RAM does not depend upon the location of the information because RAM is not sequential, hence the term random access.

Most small microcontrollers provide very little RAM which forces you to write applications that use RAM wisely. Manipulating large data structures and using pointers, re-entrant or recursive functions use large amounts of RAM and are techniques which are generally avoided on
microcontrollers. Some C instructions which are rarely used on larger platforms are more commonly used in C programs for microcontrollers. One example is the goto instruction reviled by traditional C programmers. While goto is rarely used on larger platforms, in embedded system programming it can sometimes be used to save RAM. If your hardware supports a stack, the stack contents and the area that is needed for stack management is generally allocated through the RAM. A stack is a structure which records the chronological ordering of information. It is used mainly in subroutine calls and interrupts servicing. A stack is a LIFO (last in, first out) structure. Let us take an example of how the stack is managed in a Motorola microcontroller MC68HC705C8. Here there are 64 bytes that occupies the space from address 00C0 to 00FF:

0x00C0

Stack pointer

bit number 5 4 3 2 1 0

0x00FF 0 0 0 0 0 1 1 address

Figure 7: MC68HC705C8 stack

The stack pointer contains the address of the next free location on the stack. On reset the stack pointer for the MC68HC705C8 holds the value 00FF. When the data is pushed on to the stack this decrements the stack pointer and when the data is popped then there happens and increment in the stack pointer

3.3 I/O Ports
There are two main port types, parallel and serial, and two port modes, synchronous and asynchronous. Parallel I/O requires a data line for each bit, while serial I/O uses a single line and transfers bits in sequence. Synchronous I/O is synchronized to a clock while asynchronous I/O is not. Microcontrollers most often have parallel I/O capability built in and serial I/O as a peripheral feature.

Let us go through some of the port configurations.

1. Microcontroller like COP8SAA7 do have 4 bidirectional I/O ports with each named C, G, L and F either they can act as an I/P or O/P or they can be in tri state one. Each port has an associated configuration register and data register. It also has a MICROWIRE/PLUS synchronous serial interface.

2. The Motorola MC68HC705C8 has 3 ports A, B and C. Here they are each eight bit respectively and can act as either I/P or O/P and that depends upon the value that is carried by the data direction register. It also contains a dedicated a seven bit that remains the same I/P port named D port that is kept for programming the serial port. This device also has a SCI (serial communications interface) asynchronous serial interface and a SPI (serial peripheral interface) which both use Port D for their functions.

3. There are five ports present in the PIC microcontroller PIC16C74. PORTA, PORTB, PORTC, PORTD and PORTE. The direction is controlled by the TRIS register that is present in it. ADCON1 is used by the PORTA so as to select analog or digital configuration. The eight bit parallel slave port can be configured upon by using PORTD and PORTE. The PIC16C74 has a SSP (synchronous serial port) module which can operate both in SPI and I2C modes. The device also has SCI module Serial ports have a frequency of operation
called their baud rate. The baud rate is the reciprocal of the transmission time for each bit. For example, if the baud rate is 9600 bits/second then the transmission time for each bit is \( \frac{1}{9600} \) of a second.

While microcontrollers do not support the same sophisticated input/output functions as larger platforms, such as those in the C stdio library, they still support device input and output. The input/output channels permit the microcontroller in communicating with such peripheral devices as timers, sensors, keypads and LCD screens.

Microcontroller ports are usually memory-mapped and can therefore be used like any other memory location. Ports usually consist of 8 or fewer bits which often support tristate logic with three states: input, output or impedance with very high value. The state of being undefined or floating is denoted by the high impedance mode. Some devices only support binary logic and in those cases the bit can be defined as a combination of only two of the three states. If a port has input and output ports that are programmable then there will be some associated registers that indicates whether the data is input or output. Usually the registers which indicate the status of the input or output port are called DDR(Data direction register).

The memory mapped port locations can be reserved as the compiler never keeps it for allocating the data memory. We can use the preprocessor directive #pragma to mention the location of each mapped input and output register. As per our convenience we can also assign a meaningful mnemonic name for each input and output port. The variable names that the port are being associated with can be used to do a reading or writing to a particular input or output port. On the Motorola microcontroller 68HC705C8 the below is the two ports and their associated direction registers.
#pragma portrw PORTA @ 0x0000;

#pragma portrw PORTB @ 0x0001;

#pragma portrw DDRA @ 0x0004;

#pragma portrw DDRB @ 0x0005;

Example 1: Defining ports with #pragma directives

It is then possible to write the value AC to the port using the C command:

DDRA=0xFF;       /* ouput is set as to particular direction */

PORTA=0xAC;      /* The AC value is assigned to the port */

3.4 Timer

A timer is basically a counter that get incremented at a regular interval as per the system clock pulses. There are several different types of timers available. A timer/counter can perform several different tasks. The CPU uses the timer to keep track of time accurately. The timer can generate a stream of pulses or a single pulse at different frequencies. It can be used to start and stop tasks at desired times. A watchdog timer or computer operating properly usually checks the runaway code execution. The way by which the hardware is implemented for a watch dog timer varies according to the architecture used. Usually after the reset and a few cycles the watchdog is usually reset and this in the case of software is repeated periodically. In general watchdog timers must be turned on once within the first few cycles after reset and then reset periodically with software. We can set and program the watchdog timers as per different time-out delays. There
can be complexity in the reset sequence as sending the byte sequence to a particular port or it can be very simple as in the case of a specialized instruction. The main function of a watchdog timer is to either reset the processor when its time is out or execute an interrupt after the particular time set is over. Timer configurations vary among microcontrollers. Let us go through some sample configurations.

National Semiconductor with its microcontroller COP8SAA7 contains a 3 timers --- 16 bit T1 timer, a sixteen bit T0 idle timer and a watchdog timer in which the idle timer T0 does the role of maintaining minimum power and real time throughout the IDLE mode. For real time controls the timer Ti is used. It is associated with 3 user opt able modes.

- In the Motorola microcontroller family the MC68HC705C8 there are two timers one a counter having sixteen bits and a watch dog timer COP.

- There are 4 timer modules in the case of PIC17C42. They are named as TMR0 through TMR3 also it contains a watchdog timer. Out of these timers the tmr0 is a sixteen bit timer and contains a programmable pre scalar. And tmr1 and 2 are eight bit each and again tmr3 is 16bit timer.

3.5 Interrupt Circuitry

An interrupt is an event that keeps off all the usual program operations when it is called and keeps the CPU vacant for the operation that needs the priority or that is mentioned in the interrupt. Interrupts increase the response speed to external events. Different microcontrollers have different interrupt sources which can include external, timer and serial port interrupts. When an interrupt is received current operation is suspended, the interrupt is identified and the controller jumps (vectors) to an interrupt function routine. Basically the interrupt can be imposed
Hardware-wise and software-wise. Hardware interrupts include a signal to a pin, timer overflow, and serial port interrupts. Software interrupts are commands given by the programmer, such as the SWI instruction for the Motorola MC68HC705C8.

There are two different interrupt types: maskable and non-maskable. A maskable interrupt can be disabled and enabled while non-maskable interrupts cannot be disabled and are therefore always enabled. The vector arbitration interrupts are used by almost all eight bit microchips. Vectored arbitration means that when a specific interrupt occurs the interrupt handler automatically branches to an address associated with that interrupt. The servicing of interrupts in general is dictated by the status of the GIE (Global Interrupt Enable). GIE is cleared when an interrupt occurs and all interrupts are delayed until it is set.

3.6 Buses

A bus carries information in the form of signals. There are three main buses: address, data, and control.

1) The address bus is unidirectional and carries the addresses of memory locations indicating where the data is stored. The total number of memory locations is determined by the number of wires that are being used in the address bus. With a 13 bit address bus, for example, there would be 2^13 or 8192 memory locations.

2) The data bus is bi-directional and carries information between the CPU and memory or I/O devices. Computers are often classified according to the size of their data bus. The term “8-bit
“microcontroller” refers to a microcontroller with 8 lines on its data bus. The number of wires in the data bus determines the number of bits that can be stored in each memory location.

3) The control bus carries data which controls system activity. Often this data includes timing signals which synchronize the movement of other information.

Devices controlled by Embedded systems:

- The embedded system finds a large number of applications both which finds applications for both domestic and industrial purposes. Among the domestic purpose embedded devices the main one used are the one used for automation purpose like automatic control of light systems, then controlling the volume, sound and channels of the entertainment system like TV or audio or
video systems. The embedded systems are also used for the purpose of automatic security systems which can monitor, record and intimate the owner of any illegal entry. Then its also used in automatic systems that control the flow of water in bathroom then automatic control of water heaters and other equipments. In some cases they are also used in kitchen where smoke and fire detection if found can be controlled automatically, the embedded systems are also used in automatic control of Airconditions and microwave settings. Even in the garden for water sprinklers and other purposes the embedded systems are used. Thus with the application of embedded systems a large number of domestic works can be employed with minimum human interference and with maximum power savings.

With the application and introduction of communicating systems with embedded systems like mobile there are large number of uses and advantages have been achieved in modern time they are from the mobile itself we can view the entire caller identification and the whole of the communication process when the mobile is connected with the TV, then according to the calls coming the volume of the other audio and video systems can be controlled with pre-settings. One can turn on and off the VCR from the mobile itself. Every thing can be set with the particular time allocation and time bound activities. It its rainy season the water for the garden can be controlled automatically that is the system can sense the weather and act accordingly. In case of alarm system failure the user or the driver will be intimated automatically without human interference.

Real-time applications of the embedded systems
With the fast growing development in the field of embedded systems the time required by the microcontroller to act to any external events has been reduced to micro and nano second, so as a result of the very high fastness and accuracy in handling things the embedded systems are used for the control of external events which are very sensible and of utmost importance. And this purpose has been achieved by the systems with cent percent result. From car automation where some real time control is needed and has been achieved, like the breaking system where the action need to be taken within one by tenth of a second or systems like furnace control like iron ore separation, where the different grades of iron are distilled and separated to get thousands of grade and varieties. Since it is temperature and time dependent the automation using embedded systems has achieved wonderful results.

Other applications of real time embedded systems are the use in satellite communications, missile and rocket launching systems, airborne systems, navigations, and other military purposes where hundred percentage result need to be achieved without fail.

**Over view of Software**

In embedded systems the micro controller acts as the main brain and backbone of the system, Each controller is having its own set of instruction through which the developer can program it in the way he wants to run it. It will also be aided by an assemble for converting the high level instructions or programs into machine language either it will be a dedicated one or in some cases there will be a cross assembler which will be compatible for some number of processors. This is because the assembly level programming involves the knowledge of the deep instructions and is very intricate and
need high level of dedication and knowledge of the complete system and operating system.

3.7 C programming

C is a very powerful procedure oriented language and is very useful in programming at higher level and at lower level. With the intention of utilizing and considering this as the next step and procedure this language came up and has been designed very close to microcontrollers instruction sets. So most of the commands and library functions will be resembling the instruction set type programming. The accuracy and perfection that it provides makes it the best and very apt even to design an operating system. Even some language take c as the intermediate language and gets translated into c. Certain operating systems like sun solaris makes use of lakhs of lines of c and thousands of lines in assemblers to aid its performance. C has an inbuilt standard library and from where mostly used library functions are stored and can be retrieved at easy when and wherever needed. For almost all the embedded requirements c can be efficiently used but in places where a call stack need to be used and it demands to be called in empty conditions whereas a c starts executing only from the main function so it cannot run and initialize stack call. Cross compiler support do exist for almost all the micro controllers so once u program in c it can be taken to any architecture provided the configurations alone need to be changed. Another drawback or weakness is that it can take only second position when the performance or speed between this and the handwritten assembly program runtime is compared. Then context switching cannot be done in c that is according to the privilege
and priority it cannot switch and operate. The space consumption will be very high in C as compared to the assembly one and this affects the performance when used in real time applications.

C/Assembler interface

- In C a stack is used for most of the function manipulations like when a call is made a stack frame having the information like the procedure call locations the base of the previous stack frame and the function call’s code address. It also stores the space for variable which are local in nature. The assembler method called need to understand the right format of this stack frame in order to do manipulations.

C Stack Frame

Consider a function

```c
int S(int a, int b, int c)
{
    int local1, local2, local3; [72]
    ....
}
```
Whenever a call is made the stack is filled up with the arguments involved and spaces are allocated for the local variables that are declared. The frame pointer keeps the values of the arguments the return address needed and the local variables etc. when the particular task is over while returning the stack frame is removed and the pointer is again kept.

Object oriented programming language C++

After developing a procedure oriented language like C the next development was an advanced version of C that is C++ which is an object oriented language having can the compatibility with C that is one can compile C programs in C++ environment. Almost all the library functions are available in C++ that help the C program to execute at ease and providing higher end compatibility. The object oriented approach makes the C++ language very safe and provide data security and encapsulation, though the programming is little complex.
C language and Higher level language - Java

To make it compatible and interface the java with C code there are two mechanisms which are well defined in the first mechanism there are a large volume of codes that are stored in the java virtual machines. Almost all the AWT are being developed in C and with this interface it is executed. The second was a flop and never get in to the market that is the JNI and was denied of authorization being legalized. A permanent pre assigned format is being followed by the methods in C that are being called from java. These functions work upon the objects in java. The array bound check security arrangements that are available in the java are not found in C coding so there can be a large number of bugs visible in C. With the help of java network interface the code that is in java can call a code in c which can call a code in assembly level language.[98]
3.8 Java virtual machine

The above depicts the whole of Java virtual machine architecture. Java byte code is got through the compilation of source code that is cross checked and then compiled or interpreted which intern can suite the architecture which is native. The Java Application program interface and Java Virtual Machine together makes the java runtime environment also denoted by JRE. It is in the java virtual machine that the Java byte code is run and which acts as the component where the code is executed. It is estimated that there are over five billion Java virtual machine enabled systems by Sun microsystems.
A Java virtual machine is a software fundamentally which runs on virtual and non virtual H/W on all most all the standard OS. A java virtual machine arranges a platform where we can run and execute Java byte code, and enabling special attributes like automatically handling the exceptions by which the main debugging is done with ease thereby reducing the errors in the program or reducing the exceptions and is unbiased of the source code. A Java Virtual Machine takes the help of the number of standard class libraries that are available and with which the java application programming interface is implemented. Application Programming Interface joined along with java virtual machine together make the Java Runtime Environment.[51]

The java virtual machine is being utilized by various H/W platforms and S/W applications. The use of the one single byte code for more than one java virtual machine in different platforms makes the java a very powerful and well known as program one time and run anywhere that is compile anywhere that strongly conveys it as a cross platform compiling language. So the java virtual machine forms an integral part in the case of java platform.

Though Java byte code which can be considered as language which is intermediate generally compiled at the Java and also we can compile from other programming languages too. For example the source code in Ada can be converted to java byte code by compiling it and can be easily run on a java virtual machine.

JVM was produced initially by Oracle the owner of Java, but JVMs developed by other firms by taking the java trademark provided they need to follow the oracle specification for java virtual machine that they put forward.

**Execution environment**
The Java Runtime Environment or JRE is the execution environment developed by Oracle. Programs that need to be run on a Java Virtual Machine should be converted to binary format which is portable and is of standard one that is in the .class file format. Now for the convenience of management the developer can divide the program into a group of files. A program may be composed of many classes in different files. To distribute in a quite easy way large programs are made as class files which are multiples and they can also be clubbed together to form a .jar file or that is for Java archive format.

JAVA application launcher in the case of java provides a standard format for running Java code. The runtime for java runs and execute .class or .jar files, the instruction set in java virtual machine is emulated with the interruption of it or by utilizing the compiler that is of the type just in time one. The just in time compiler is applied in almost all the java virtual machines today so as to attain the efficiency and improved speed. Here the time compilers are also used that to aid the design engineers to precompile class files into native code so as to make it suite the each and every platforms applicable.[126]

As with other virtual machines, the stack based architecture is being followed by Java virtual machine based upon the microcontroller used. The java virtual machine too support the classes and functions and thereby make it a highly optimized model with high level capability and unparallel architecture

**Java Virtual Machine languages**

Though initially the intended use of JVM is to compile and run programs in java now it also support more language to run above it. Languages that are dynamically typed are not supported
through the JVM built in feature. The existing Java virtual machine instruction set is typed statically and also used in dynamic languages by implementing the interpreters. The java virtual machine contains a restricted support to modify during the runtime itself the classes and methods that exist before Java 7. only in those environment where debugging is required it can be employed that is in places where there is a possibility to change during the running time the classes and functions.. But more superior it provide support for languages that are dynamic in Java Virtual Machines from Java 7 onwards.

**Byte code verifier**

The basic concept and theme of Java is inherently it is very secure from the security aspects and no general program that was written will do any harm or fail or corrupt the host machine or cause interrupt which is not proper with any other functions on the host machine, so one can take care of the methods and the data structures that belong a section of code that are reliable from any kind of unwanted access or corruption by reliable code that runs inside the same java virtual machine. Usual programmer errors that usually cause many problems like damaging and corrupting the data or behaving in a most unexpected way like unwantedly accessing the data from an array or making use of an pointer which is never initialized should not be encouraged at all. Many properties of the java join together to make it very safe some of the attributes like the class model, collecting the garbage the inclusion of verifier makes it much safer. [10]

Before the execution the Java Virtual Machine always cross checks all the byte code. There are basically three checking types that do exist here.

- Always valid locations are made to all the branches
- Always the initialisation and references are made to the data
- Unauthorized and unprivileged cannot access the private data and functions.

During verification process the first and the second check happens during the loading of class takes place and makes it usable. The third check takes action whenever any data or function is being accessed by some unauthorized class.

The verifier allows only certain sequence of byte code in a standard program example the jump instruction or an instruction stating the branching will be able to influence within a function. Again it guarantees that instruction that is stated do act upon only on a predefined stack location there by permitting the just in time compiler to get changed into fixed register accesses the access to stack. When using a JIT compiler on architectures that are register based it does not encourage emulation as the java virtual machine follows stack type model. In code verified JVM architecture one, the JIT compiler doesn’t feel the difference even though it is visualised as virtual register type or having stack positions that are imaginary and is expected to share this with the intended model’s registers. The code cross check process makes the java virtual machine more apt and suitable from a standard stack architecture with the point of view that to carry out emulation in a successful way using a just in time compiler is very much tedious and is generally carried out an interpreter which is rated low in performance.

The verification of code makes sure that bit patterns that are taken arbitrarily are not allowed to be used as an address. Even without a memory management unit or simply MMU memory protection is achieved. Even without the employment of additional memory management unit there are simple architecture through which the protection can be achieved in a java virtual machine. Here it is analogous to the codes in Microsoft which are managed in .NET common
language runtime and in concept nearly common to plessey 250 and IBM systems which are capability architectures.

In byte code verifier original specifications used natural language that were nor complete or correct in certain aspects. To make the java virtual machine as a standard one a series of attempts have been made. In the java virtual machine implementation aspects the security can be widely in depth and thoroughly can be analyzed thereby making all the potential security bleach and exploits can be prevented and the possible potential security exploits can be prevented and safeguarded. Then it is very feasible to optimize the java virtual machine by neglecting the safety checks that are unwanted and waste in the case that the running application is proved to be safe.

**Byte code instructions**

The tasks that have instructions in java virtual machine are:

- Loading and storing
- Conversion of data types
- Arithmetic
- The formation and manipulation of object
- Managing the operand stack i.e. push or pop etc.
- Branching or control transfer
- Invoking methods
- Handling exceptions.
- Concurrency based upon monitor.

To achieve the binary compatibility is the aim. So different operating systems demands their own java virtual machine implementation and its runtime. The byte code that is got through the java virtual machine interpreted one semantically will be one but the one with really implemented will be entirely different. Here rather than the complexity of just emulating byte code is compatibly and implementing efficiently the application program interface for the core java so that it can link itself with other operating systems.[20]

Heap

The memory storage used by the java virtual machine is termed as the heap and is also called the Hotspot. It aids in the dynamically allocating memory. It is classified as generations. First is the young generation where the objects that are short lived are created and immediately collected in the garbage are stored.

The old generation holds the objects that remain for longer time. This generation is also termed as the tenured one.

For defining the class and associated meta data the permanent generation is used.

Only in the same area both the objects and classes were stored together and there is no permanent generation. There is no generation that is fixed and permanent one the classes and objects are put together and kept in the same storage place. When the unloading of class happen
the objects are gathered and to the particular area the class structures are kept and this improves the performance significantly. [127]

Secure execution of remote code

The code inside the machine that is which resides within it are allowed to take the control of very fine grained over the action inside it. From the remote sources the most unreliable and trustworthy codes are allowed to execute safely and is a model by Applets. Inside a virtual machine that is within a users browser the applets run and the code downloaded from a remote HTTP server is being executed here. [136] To protect the user from unwanted and misbehaving codes that can create problem a restricted sandbox is used to run the remote code. To access the local files from the system or clipboard or network one has top get the permission to break out of the sandbox for which a certificate which is digitally signed applets that are safe are to be purchased by the publishers.

JAVA CARD TECHNOLOGY

Java card Platform helps smart cards and many other low-cost equipments having very restricted memory space. That is even with 8KB RAM to run small Java applications. This feature enable smart card manufacturers with an execution platform that is safe security wise and can be operated from within a single device and which will be able to store and update a number of applications in it. Certain applications for java card include SIM cards which are used in mobile phones and other wireless N/Ws it is also used in bank debit and credit cards to do online and other transactions. The identity cards issued by the government and other health department too finds its application.
Java Embedded Technologies Development Tools

For Java embedded technologies a wide range of development tools promises and enables over nine million Java developers to be productive and well-supported for their applications while editing, compiling, and debugging. There are a wide range for the developers to select from IDE’s with industry-standard including Net Beans and Eclipse, or tools like Java Platform of Oracle, Java ME Software Development Kit or SDK that are micro editions are targeted tools which provide a full development environment for CDC and CLDC-based systems. Here the developer can do writing, editing, compiling, package, sign, obfuscate the applications, do testing and debugging on Windows or Mac OS X desktops device emulators and built-in profiling support with the support of Software Development Kit of java millennium edition.[34] The Java ME Software Development Kit also incorporates monitor networking and memory usage ability which in fact enables developers to produce better quality, higher performance applications.

Embedded Database and Synchronization Support

Software giants like Oracle promises a high performance, transactional database that perfectly suited to both Java SE Embedded and CDC based platforms. A collection of Java and native language modules that are products of Oracle Berkeley DB Java family that promises developers full support for concurrent, transactional, and highly-available storage.[111] Embedded data stored in devices running Berkeley Database to be synchronized with Oracle's enterprise products is enabled with the help of database mobile Server. It also helps in managing the provisioning and lifecycle on device applications.