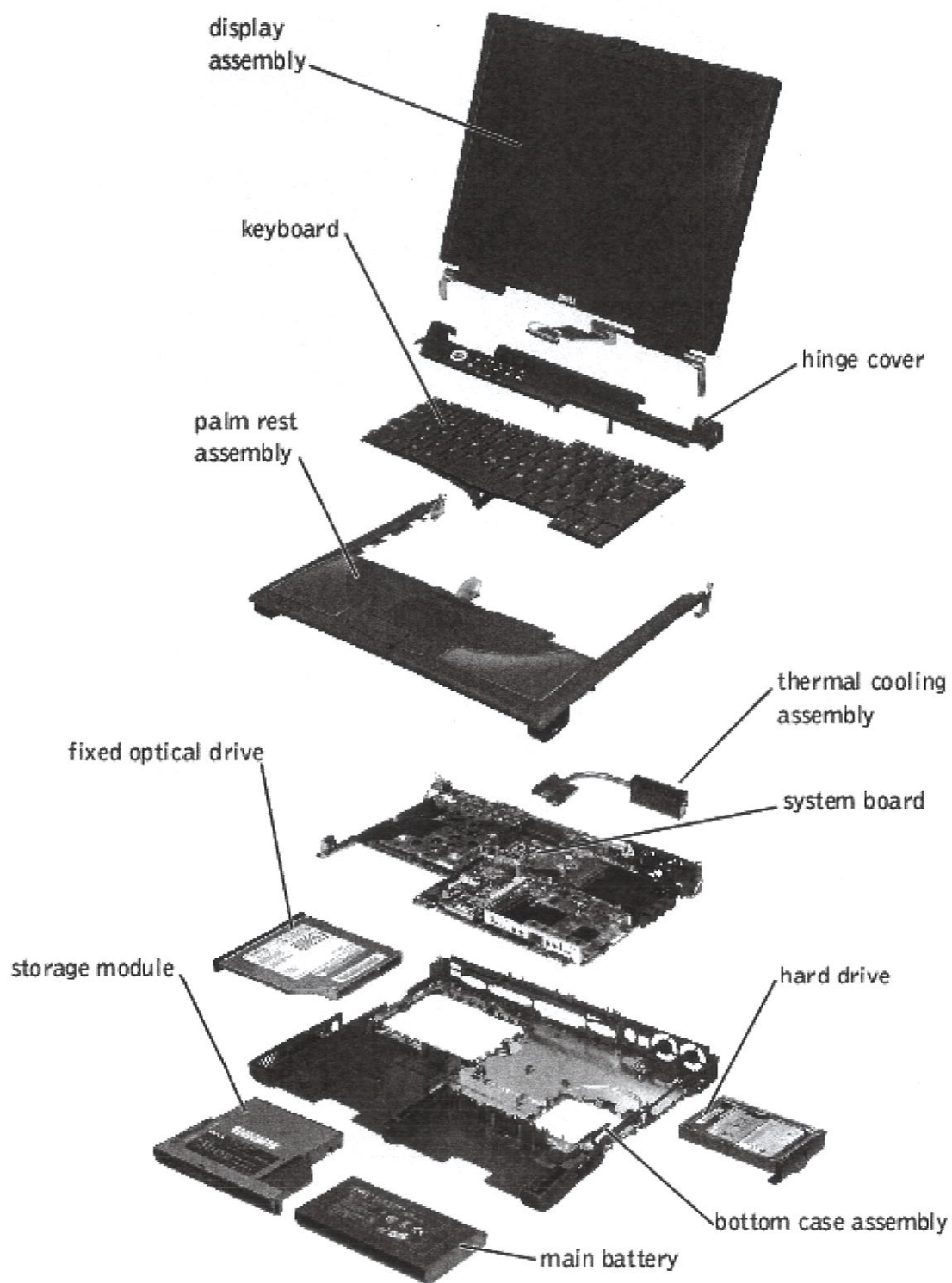
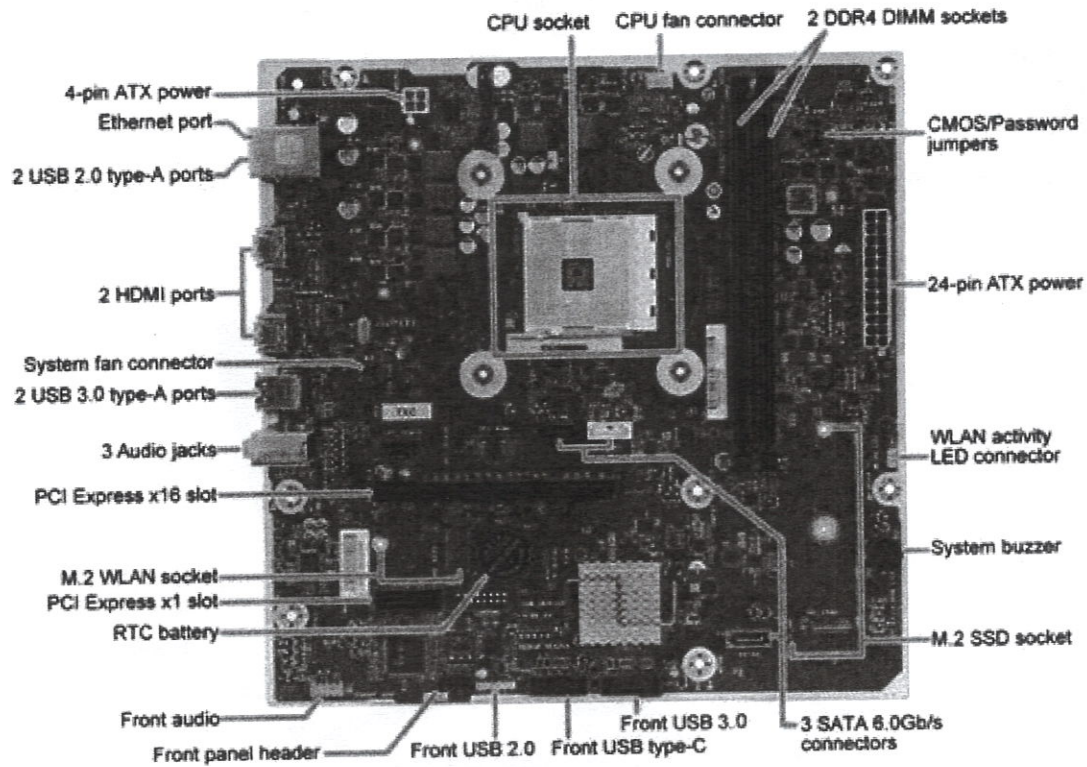




Temasek Junior College
H2 Computing
Computer Architecture

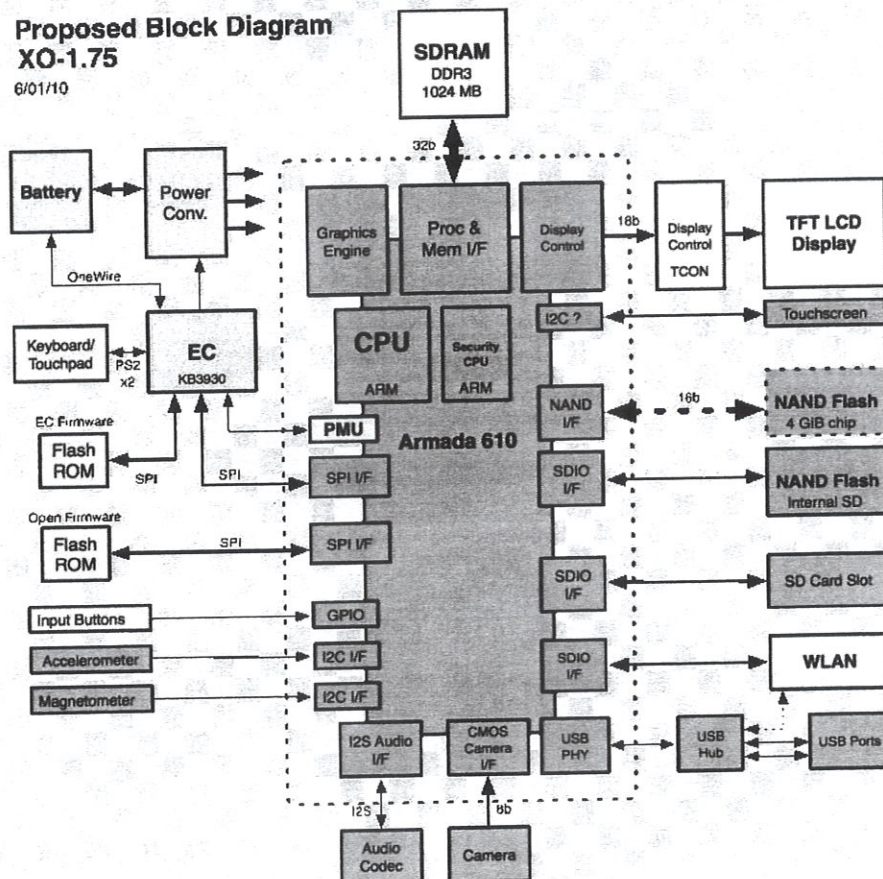
Know your system:

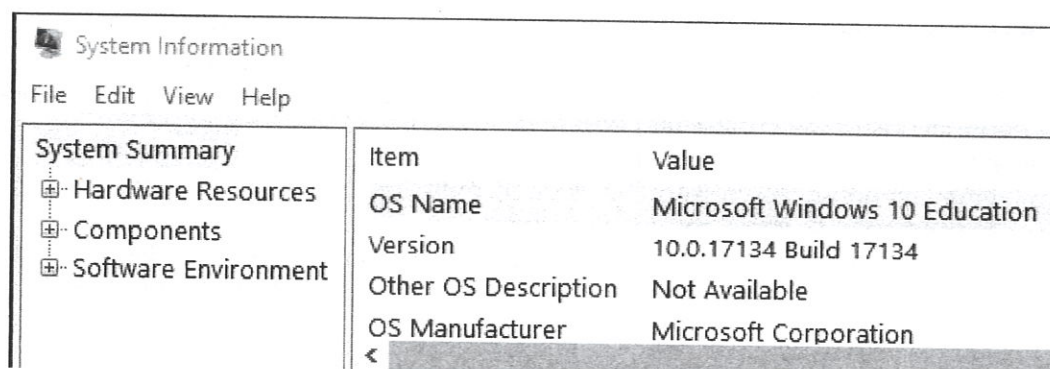
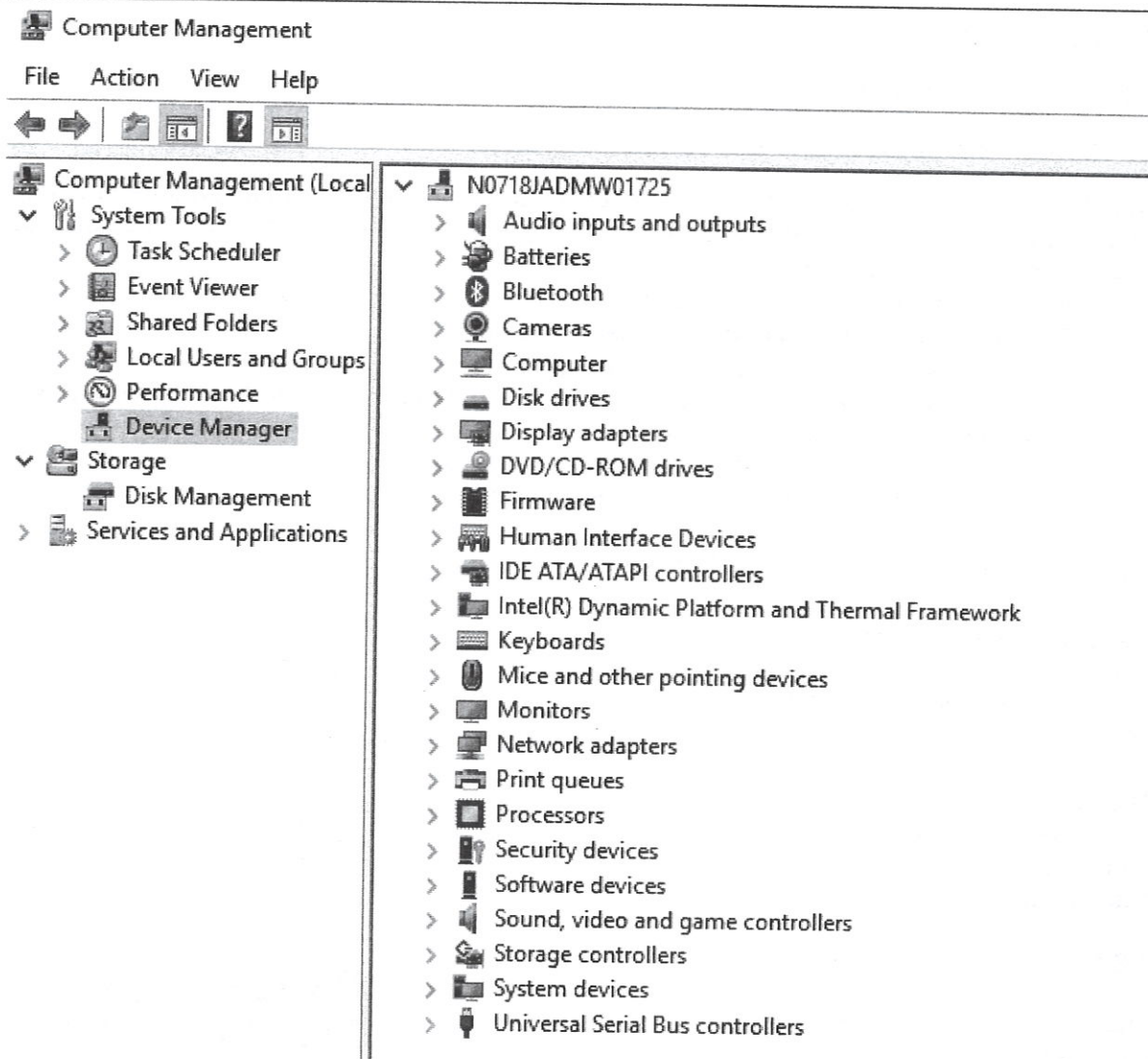




Proposed Block Diagram XO-1.75

6/01/10





Computer Architecture

1. Processor Fundamentals

1.1 The Von Neumann model of a computer system

The simplest form of what might be described as a computer system architecture is usually attributed to John von Neumann. This recognises the fact that he was the first to describe the basic principles in a publication.

The model has the following basic features:

- There is a processor, a central processing unit.
- The processor has direct access to a memory.
- The memory contains a 'stored program' (which can be replaced by another at any time) and the data required by the program.
- The stored program consists of individual instructions.
- The processor executes instructions sequentially.

1.2 Central processing unit (CPU) architecture

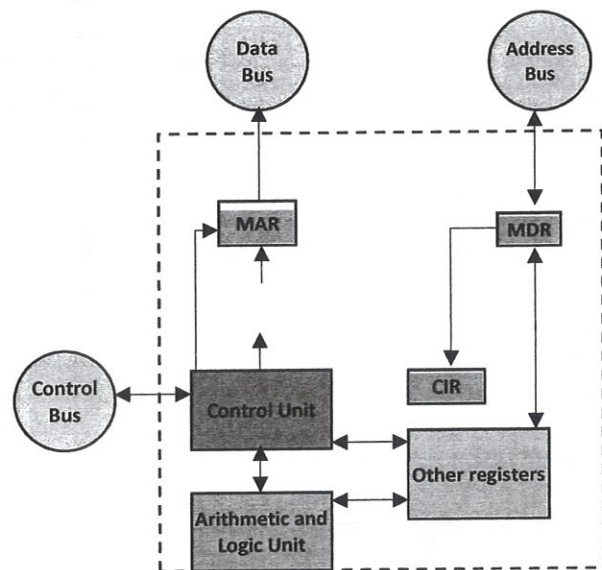
The dotted outline shows the boundary of the processor. The logical arrangement of some of the processor components is indicated. The arrows show possible directions of flow of data.

The data for some of the arrows is specifically an address or an instruction. However, in general, data might be an instruction, an address or a value.

1.2.1 Components of the CPU

The two major components of the CPU are the arithmetic and logic unit (ALU) (or Arithmetic Logic Unit) and the control unit.

- ALU is responsible for any arithmetic or logic processing that might be needed when a program is running.
- The functions of the control unit are more diverse.
 - One aspect is controlling the flow of data throughout the processor and, indeed, throughout the whole computer system.
 - Another is ensuring that program instructions are handled correctly. A vital part of the control unit is a clock which is used by the unit to synchronise processes.



A schematic diagram of the architecture of a CPU

1.2.2 Registers

These are storage components which, because of their proximity to the ALU, allow very short access times. Each register has limited storage capacity, typically 16, 32 or 64 bits. A register is either general purpose or special purpose. If there is only one general-purpose register it is referred to as the accumulator.

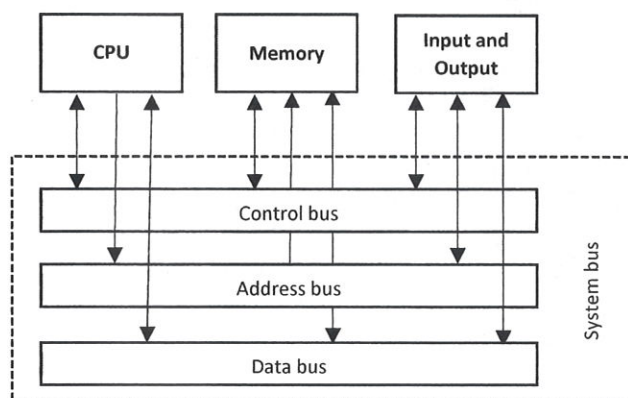
Register name	Abbreviation	Register's function
Current instruction	CIR	Stores the current instruction while it is being register decoded and executed
Index register	IX	Stores a value; only used for indexed addressing
Memory address register	MAR	Stores the address of a memory location which is about to have a value read from or written to
Memory data register (memory buffer register)	MDR (MBR)	Stores data that has just been read from memory or is just about to be written to memory
Program counter	PC	Stores the address of where the next instruction is to be read from
Status register	SR	Contains individual bits that are either set or cleared

1.3 The system bus

A bus is a parallel transmission component with each separate wire carrying a single bit. Instead it is a mechanism for data to be transferred from one system component to another.

System bus comprises three distinct components:

The address bus, the data bus and the control bus.



A schematic diagram of the system bus

- the address bus is connected to the MAR
 - The sole function of the address bus is to carry an address. The address specifies a location in memory which is due to receive data or from which data is to be read.
 - The crucial aspect of the address bus is the '**bus width**', which is the number of separate wires in the bus. The number of wires defines the number of bits in the address's binary code. In the simple computer system considered here we will assume that the bus width is 16 bits allowing 65 536 memory locations to be directly addressed.
- the data bus is connected to the MDR

The function of the data bus is to carry data. This might be an instruction, an address or a value. The data bus might be carrying the data from CPU to memory or from memory to CPU.
- Bus **width** is again an important factor in considering how the data bus is used. Before discussing this, it is useful to introduce the concept of a **word**. A word consists of a number of bytes and for any system the word length is defined. The significance of the word length

is that it defines a grouping that the system will handle as one unit. The word length might be stated as a number of bytes or as a number of bits. Typical word lengths are 16, 32 or 64 bits that is, 2, 4 or 8 bytes respectively. For a given computer system, the bus width is ideally the same as the word length.

- the control bus is connected to the control unit.
The control bus is another bidirectional bus which transmits a signal from the control unit to any other system component or transmits a signal to the control unit.
- The system bus allows data flow between the CPU, the memory, and input or output (I/O) devices.

1.4 The fetch-execute cycle

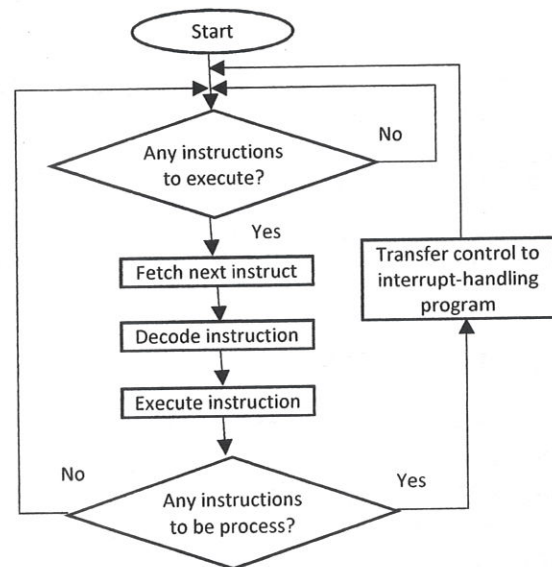
Fetch, decode and execute cycle.

If we assume that a program is already running then the program counter already holds the address of an instruction. In the fetch stage, the following steps happen:

- This address in the program counter is transferred within the CPU to the MAR.
- During the next clock cycle two things happen simultaneously:
 - the instruction held in the address pointed to by the MAR is fetched into the MDR
 - the address stored in the program counter is incremented.
- The instruction stored in the MDR is transferred within the CPU to the CIR.

For our simple system the program counter will be incremented by 1. However, it should be noted that the instruction just loaded might be a jump instruction. In this case, the program counter contents will have to be updated in accordance with the jump condition. This can only happen after the instruction has been decoded.

In the decode stage, the instruction stored in the CIR is received as input by the circuitry within the control unit. Depending on the type of instruction, the control unit will send signals to the appropriate components so that the execute stage can begin. At this stage, the ALU will be activated if the instruction requires arithmetic or logic processing.





Flowchart for the fetch, decode and execute cycle

2 Hardware

2.1 The memory system

A simplified version of a memory system hierarchy to trends in the important factors affecting this choice. The factors increase in the direction of the arrow.

Component	Category	Access time	Capacity Size Cost
Register	Processor component		
Cache memory	Primary storage		
Main memory			
Hard disk	Secondary storage		
Auxiliary storage			

Trends in the factors affecting the choice of memory components

2.2 Memory components

The processor has direct access to three types of storage component. The registers are contained within the processor. External to the processor there is cache memory and main memory, which together constitute the primary storage. Cache memory is used to store data that at any time is the most likely to be needed again by the processor.

- Random-Access Memory (RAM).

RAM can be accessed at any location independently of which previous location was used (it might have been better called 'direct access memory'). A better description is read-write memory because RAM can be repeatedly read from or written to. Another distinguishing characteristic of RAM is that it is volatile which means that when the computer system is switched off the contents of the memory are lost.

- Read-Only Memory (ROM).

ROM shares the random-access or direct-access properties of RAM except that it cannot be written to.

- Memory management

Code Segment

In computing, a code segment, also known as a text segment or simply as text, is a portion of an object file or the corresponding section of the program's virtual address space that contains executable instructions. The term "segment" comes from the memory segment, which is a historical approach to memory management that has been succeeded by paging. When a program is stored in an object file, the code segment is a part of this file; when the loader places a program into memory so that it may be executed, various memory regions are allocated (in particular, as pages), corresponding to both the segments in the object files and to segments only needed at run time.

Data segment

A data segment is a portion of memory which contains initialized static variables, that is, global variables and static local variables. The size of this segment is determined by the size of the values in the program's source code, and does not change at run time. The data segment is read-write, since the values of variables can be altered at run time.

BSS - Block Started by Symbol

The BSS segment, also known as *uninitialized data*, is usually adjacent to the data segment. The BSS segment contains all global variables and static variables that are initialized to zero or do not have explicit initialization in source code.

Heap

The heap area commonly begins at the end of the .bss and .data segments and grows to larger addresses from there.

Stack

The stack area contains the program stack, a LIFO structure, typically located in the higher parts of memory. A "stack pointer" register tracks the top of the stack; it is adjusted each time a value is "pushed" onto the stack. The set of values pushed for one function call is termed a "stack frame". A stack frame consists at minimum of a return address. Automatic variables are also allocated on the stack.

The stack area traditionally adjoined the heap area and they grew towards each other; when the stack pointer met the heap pointer, free memory was exhausted.

