Automotive & Embedded Info

Never Forget Basics Whether its Life or Anything Else ... Basics are Cores. While seeing a Tree how can we forget Seed...

Basic Embedded Info like UC, Memory etc.

8, 16 & 32 bit Controller: –

1. Controller is 8, 16 or 32 bit, it depend on data bus of the controller.
2. No of data will be different for 8, 16 & 32 bit processor.
3. 8, 16, 32 bit will do 8, 16, 32 bit mathematical operation respectively.

Microprocessor VS Microcontroller: –

A microprocessor generally does not have Ram, ROM and IO pins. It usually uses its pins as a bus to interface to peripherals such as RAM, ROM, Serial ports, Digital and Analog IO. It is expandable at the board level due to this.	A microcontroller is 'all in one', the processor, ram, IO all on the one chip, as such you cannot (say) increase the amount of RAM available or the number of IO ports. The controlling bus is internal and not available to the board designer.
It is just a processor. Memory and I/O components have to be connected externally	Micro controller has external processor along with internal memory and i/O components

Mainly used in personal computers	Used mainly in washing machine, MP3 players

Harvard & Vonueman: –

VONUEMAN	**HARVARD**
Same memory holds data, instructions.	Separate memories for data and instructions
A single set of address/data buses between CPU and memory	Two sets of address/data buses between CPU and memory
	Harvard allows two simultaneous memory Fetches.

RISC & CISC: –

RISC reduced instruction set computer	**CISC complex instruction set computer**
1. Emphasis on Software	1. Emphasis on Hardware
2. Single clock, reduced instruction only.	2. Include multi-clock complex instructions
3. Small set of instructions(32 instruction set)	3.Very large instruction sets up to three hundred separate instructions.
4. Large code size,	4. Small code size,
5. Spends more Transistors on memory registers.	5. Transistor used for storing complex instruction.
6. Register to Register: LOAD and STORE are independent instructions	6. Memory to Memory: LOAD and STORE incorporated in instructions.

7. Mainly used in real time application.	7. Mainly used in normal pc, workstations and servers.
8. Example: ARM, PIC, ATMEL AVR.	8. Example: 8085, x86, x64 from intel and 8051 from ATMEL
9. In RISC, instructions are simple i.e. single instruction can performs a simple (generally single) function e.g for AVR micro-controller "DEC reg"(decrease reg by 1) "BRNE k" (if not equal to zero, it adds k to the program counter).one instruction is performing one operation. So, two instructions are needed for performing the same operation in AVR.	9. In CISC, instructions are complex i.e. single instruction can performs a complex function. E.g "DJNZ reg/mem, addr" (decrease and jump if not zero) instruction in 8051 micro-controller decreases the value in register "reg" or memory location "mem" by 1 and then check if it is equal to zero. If the value is not zero, it jumps to given address. (it adds the "addr" to the program counter)
10.LOADA,2:3 LOADB,5:2 PRODA,B STORE 2:3, A	10.MULT 2:3, 5:2

Bitrate & Baud rate: –

The difference between bit rate & baud rate is complicated &intertwining. Both are dependent and inter related.

Bit rate is how many data bits are transmitted per second. A figure of 2400 bits per second means 2400 zero & ones can be transmitted in one second.

A baud is number of times a signal in a communication channel changes state or varies. A 2400 baud rate means that the channel can change state up to 2400 times per second. The term change state means that it can change from 0 to 1 or from 1 to 0 up to x (in this case, 2400) times per second.

Bits per second = baud per second * number of bit per baud.

Interrupt: –

Interrupt stop the continuous execution of the process or activity. Software interrupt – timer etc. And hardware interrupt – serial, external etc.

Interrupt Latency: –

Also called interrupt response time, is the length of time that it takes for an interrupt to be acted on after it has been generated. Factor that affect interrupt latency is microprocessor design (architecture), the microprocessor clock speed, the particular OS employed and the type of interrupt controller used. Minimum interrupt latency depends mainly on configuration of the interrupt controller, which combines interrupts onto the processor lines and assigns priority level to the interrupts. Maximum interrupt latency depends mainly on OS.

Do not pass any o/p function and data processing in ISR.

Watchdog Timer: –

It is an electronic timer that is used to detect & recover from UC malfunction. The controller regularly starts the watchdog timer to prevent it from elapsing to timing out. If due to hardware fault or program error, the computer fails to restart the watchdog timer, the timer will elapse and generate a timeout signal or reset the processor.

Shift Register: –

They are group of flip flop connected in a chain so that the o/p from one flip flop becomes the I/P of the next flip flop. All flip flops are driven by one common clock, all are set and reset simultaneously. Types of shift register à 1. Serial-in serial-out 2. Serial-in parallel-out 3. Parallel in serial out 4. Parallel in parallel out.

Bit Banging:

Bit banging is technique for serial communication using software instead of dedicated hardware. Software directly sets and samples the state of pins on the microcontroller and is responsible for all parameter of the signal: timing, level, synchronization etc.

Multi core Processor: –

A multi-core processor is a single computing component with two or more independent actual processing units (called "cores"), which are the units that read and execute program instructions. The instructions are ordinary CPU instructions such as add, move data, and branch, but the multiple cores can run multiple instructions at the same time, increasing overall speed for programs amenable to parallel computing.

A central processing unit (CPU) is the electronic circuitry within a computer that carries out the instructions of a computer program by performing the basic arithmetic, logical, control and input/output (I/O) operations specified by the instructions.

Compilation & Its Stages: –

Compilation is a process of converting source code to object

Stage of compilation: -pre-processing, compilation, assembly & linking.

Pre-processor: process all pre-processor command written in high level language before compiler takes over.

Syntax Check: Syntax check happens here.

Compiler: converts c source code/files (high level language) into object code/files (low level language). Compiler is set of program that transform source code in to another computer programming language (target language, having a binary form known as object code).

Assembler: converts assembly code/files (low level language) into object code/files.

Linker: converts object code/file into executable code/file (hex file etc). Linker is a set of programs that takes one or more object files generated by compiler and combines them into a single executable/library or another object file.

An object file can contain three kind of symbols: 1. Defined external, 2. Undefined external & 3. Local symbol.

1. It allow it to be called by other module.
2. It reference other module where these symbols are defined.
3. Used internally within the object file to facilitate relocation.

Concept of Boot Loader: –

Picture this scenario: you have a fair amount of storage on your microcontroller – enough to store more than 2-3 programs or applications that are independent of each other. Suppose that when you boot your device, you may want to be able to choose which one to run. So what would you need to support this? You would need a starting program which then allows you to choose between the others at boot-time.

A boot loader is that program – it is the first thing to run and can load other applications into specific places in memory (either persistent like flash, or volatile like ram) and then jumps to that desired program where it will then take over execution from there.

Boot Loader Steps: –

1. CPU power on then after reset it goes to defined point called reset vector.
2. Reset vector-the reset vector is the default location a CPU will go to find the first instruction it will execute after a reset. The reset vector is a pointer or address, where the CPU should always begin as soon as it is

able to execute instructions. This is usually written in assembly. It is the very first thing that runs at start-up and can be considered hardware-specific code. It will usually perform simple functions like setting up the processor into a pre-defined steady state by configuring registers and such. Then it will jump to the start-up code.

3. Start-up code- this is the first software-specific code that runs generated by your compiler/libc. Its job is basically to set up the software environment so that c code can run on top. For example, c code assumes that there is a region of memory defined as stack and heap. These are usually software constructs instead of hardware. Therefore, this piece of start-up code will define the stack pointers and heap pointers and such.it performs the following things.
4. Configure and turn on any external memory (if absolutely required, otherwise left for later user code).
5. Establish a stack pointer.
6. Clear the .bss segment (usually). Bss-block started by symbol
7. Copy from the end of .text the non-const .data.
8. Perform other static initializers.
9. Call main ().

MEMORY:

MEMORY	**RAM/DATA**	**STACK**	Execute current execution od program.
		HEAP	Used in dynamic memory allocation. It manages by realloc, calloc, malloc & free.
		BSS (UNINITIALIZE DATA)	Contain uninitialized global & static variables
		DATA (INITLIAZE	Initialize by programmer. Classified into initialized read only area and

	DATA)	initialized read/write area.char s[] = “hello world” and int debug=1 outside the main would be stored in initialized read-write area. const char* string = “hello world” makes the string literal “hello world” to be stored in initialized read-only area and the character pointer variable string in initialized read-write area.
	CPU	SFR & I/O Registers
ROM/CODE	**TEXT SEGMENT**	Contain executable instructions. Fixed size and read only.

EPROM, EEPROM & FLASH: –

EPROM “**E**“rasable “**P**“rogrammable “**R**“ead “**O**“nly “**M**“emory.

Erasable means that the chip can be erased and reused. An EPROM is erased in a device called an EPROM eraser. The eraser is a high intensity ultraviolet light source in a box. **Programmable** means that the EPROM can be programmed with a program, data or both. **Read Only Memory** means that the CPU which is connected to the EPROM can only get information from the chip. It cannot put information into the chip; thus the term ***read only***.

EEPROM: “**E**lectrically” “**E**rasable” “**P**rogrammable” “**R**ead **O**nly **M**emory”.

EEPROM: erased by exposing it to an electrical charge. The bytes in them can be selectively erased and each byte can be stored again by writing desired bytes.

FLASH: It is just one type of EEPROM

EEPROM Vs FLASH: The main difference between EEPROM and Flash is the type of logic gates that they use. While EEPROM uses the faster NOR (a combination of Not and OR), Flash uses the slower NAND (Not and AND) type. The NOR type is a lot faster than the NAND type but there is the matter of affordability as the former is significantly more expensive than the NAND type. Another advantage of EEPROM over Flash is in how you can access and erase the stored data. EEPROM can access and erase the data byte-wise or a byte at a time. In comparison, Flash can only do so block-wise. In order to simplify the whole thing, individual bytes are grouped into a smaller number of blocks, which can have thousands of bytes in each block. This is a bit problematic when you only want to read or write to a single byte at a time; which is what's typically needed in executing the code of a program. This is a reason why Flash cannot be used in electronic circuits that require byte-wise access to data. Data in Flash can also be executed, but it needs to be read as a whole and loaded into RAM beforehand. Flash is when large amounts are needed while EEPROM is used when only small amounts are needed.

Volatile & Non Volatile: –

Memory can be either volatile and non-volatile memory. **Volatile memory** is a memory that loses its contents when the computer or hardware device loses power. Computer RAM is an example of a volatile memory and is why if your computer freezes or reboots when working on a program, you lose anything that hasn't been saved. **Non-volatile memory**, sometimes abbreviated as NVRAM, is a memory that keeps its contents even if the power is lost. EPROM is an example of a non-volatile memory.

Automotive & Embedded Info / Powered by WordPress.com.