Basic interview questions on embedded systems

Introduction about Embedded systems:

An Embedded systems is a machine framework with a dedicated capacity inside a bigger mechanical or electrical framework, frequently with real time-computing constraints. It is implanted as a feature of a complete gadget often including equipment and mechanical parts. By contrast, an universally useful machine, for example, a (PC), is intended to be adaptable and to meet an

extensive variety of end-client needs. Embedded systems control numerous gadgets in common use today. here are for you the common embedded systems interview questions.

Advanced Embedded systems are frequently focused around micro controllers (i.e. Cpus with incorporated memory or fringe interfaces) however ordinary microprocessors (utilizing outside chips for memory and fringe interface circuits) are still normal, particularly in more complex systems. In either case, the processor(s) utilized may be sorts running from rather broadly useful to exceptionally had some expertise in certain class of reckonings, or even specially crafted for the application nearby. A typical standard class of committed processors is the computerized indicator processor (DSP).

The key characteristic, however, is being dedicated to handle a particular task. Since the embedded system is dedicated to specific tasks, design engineers can optimize it to reduce the size and cost of the product and increase the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.

Physically, embedded systems range from portable devices such as digital watches and MP3 players, to large stationary installations like traffic lights, factory controllers, and largely complex systems like hybrid vehicles, MRI, and avionics. Complexity varies from low, with a single microcontroller chip, to very high with multiple units, peripherals and networks mounted inside a large chassis or enclosure.

Interview question for freshers:

1. What is lst file?

- This file is also called as list file.
- It lists the opcodes ,addresses and errors detected by the assembler.
- List file is produced only when indicated by the user.
- It can be accessed by an editor and displayedon monitor screen or printed.
- Progammer uses this file to find the syntax errors and later fix them.

2. How is a program executed' bit by bit' or' byte by byte'?

EXAMPLE

ADDRESS	OPCODE	PROGRAM
1 0000		ORG 0H
2 0000	7D25	MOV R5,#25H
3 0002	7F34	MOV R7,#34H
4 0004	2D	ADD A, R5
5 0005		END

• A program is always executed byte by byte.
• Firstly,1st opcode 7D is fetched from location 0000 and then the value 25 is fetched from 0001 .
• 25 is then placed in the register R5 and program counter is incremented to point 0002.
• On execution of opcode 7F, value 34 is copied to register R7.
• Then addition of contents of R5 and accumulater takes place.
• Here all the opcodes are 8 bit forming a byte.

3. Explain DB.

• DB is called as define byte used as a directive in the assembler.
• It is used to define the 8 bit data in binary ,hexadecimal or decimal formats.
• It is the only directive that can be used to define ASCII strings larger than two characters.
• DB is also used to allocate memory in byte sized chunks.
• The assembler always converts the numbers Into hexadecimal.

4. What is EQU?

• EQU is the equate assmbler directive used to define a constant without occupying a memory location.
• It associates a constant value with data label .
• Whenever the label appears in the program ,constant value is subsituted for label.
• Advantage: The constant value occuring at various positions in a program can be changed at once using this directive.
• Syntax: label EQU constant value

5. How are labels named in assembly language?

• Label name should be unique and must contain alphabetic letters in both uppercase and lowercase.

• 1st letter should always be an alphabetic letter.
• It can also use digits and special characters ?,.,@,_,$.
• Label should not be one of the reserved words in assembly language.
• These labels make the progam much easier to read and maintain.

6. Are all the bits of flag register used in 8051?

• The flag register also called as the program status word uses only 6 bits.
• The two unused bits are user defineable flags.
• Carry ,auxillary carry ,parity and overflow flags are the conditional flags used in it.
• PSW.1 is a user definable bit and PSW.5 can be used as general purpose bit.
• Rest all flags indicate some or the other condition of an arithematic operation.

7. Which bit of the flag register is set when output overflows to the sign bit?

• The 2nd bit of the flag register is set when output flows to the sign bit.
• This flag is also called as the overflow flag.
• Here the output of the signed number operation is too large to be accomodated in 7 bits.
• For signed numbers the MSB is used to indicate the whether the number is positive or negative.
• It is only used to detect errors in signed number operations.

8. Which register bank is used if we use the following instructions

SETB PSW.3 A
SETB PSW.4 B

• Statement A sets 3rd bit of flag register.
• Statement B sets 4th bit of flag register.
• Therefore register bank 3 is initiated .
• It uses memory location 18H to 1FH.
• The register bank is also called as R3.

9. Issues related to stack and bank 1.

• Bank 1 uses the same RAM space as the stack.
• Stack pointer is incremented or decremented according to the push or pop instruction.
• If the stack pointer is decremented it uses locations 7,6,5… which belongs to

register bank 0.
• If a given program uses R1 then stack is provided new memory location.
• The push instruction may also take stack to location 0 i.e.it will run out of space.

10.Explain JNC.

• It is a command used to jump if no carry occurs after an arithematic operation.
• It is called as jump if no carry(conditional jump instruction).
• Here the carry flag bit in PSW register is used to make decision.
• The processor looks at the carry flag to see if it is raised or not.
• If carry flag is 0 ,CPU fetches instructions from the address of the label.

11.Write a program to toggle all bits of P1 every 200ms.

```
        MOV     A,#55H
AGAIN:  MOV     P1,A
        ACALL   DELAY
        CPL     A
        SJMP    AGAIN

DELAY:
        MOV     R5,#9
HERE1:  MOV     R4,#242
HERE2:  MOV     R3,#255
HERE3:  DJNZ    R3,HERE3
        DJNZ    R4,HERE2
        DJNZ    R5,HERE1
        RET
```

• Here the delay produced is 9*255*4MC*90=199,940 micro seconds.
• CPL is used to toggle the bits of P1.
• Short jump is jusd to produce a continuous loop.

12.Can port 0 be used as input output port?

• Yes, port 0 can be used as input output port.
• Port 0 is an open drain unlike ports 2,3,4.
• To use it as input or output the 10k ohm pull-up resisters are connected to it externally.

• To make port 0 as input port it must be programmed by writing 1 to all bits.
• Example:

```
MOV  A,#0FFH
MOV  P0,A
```

13. Which 2 ports combine to form the 16 bit address for external memory access?

• Port0 and port2 together form the 16 bit address for external memory.
• Port0 uses pins 32 to 39 of 8051 to give the lower address bits(AD0-AD7)
• Port2 uses pins 21 to 28 of 8051 to give the higher address bits(A8-A15)
• This 16 bit address is used to access external memory if attached.
• When connected to external memory they cannot be used as input output ports.

14. Can single bit of a port be accessed in 8051?

• Yes,8051 has the capability of accessing only single bit of a port.
• Here only single bit is accessed and rest are unaltered.
• SYNTAX: "SETB X. Y".
• Here X is the port number and y is the desired bit.
• Example: SETB P1.2

Here the second bit of port 1 is set to 1.

15. Other than SETB ,CLR are there any single bit instructions ?

• There are total 6 single-bit instructions.
• CPL bit : complement the bit (bit= NOT bit).
• JB bit,target: Jump to target if bit equal to 1.
• JNB bit,target: Jump to target if bit is equal to 0.
• JCB bit,target: Jump to target if bit is equal to 1 and then clear bit.

16. How does, taking the address of local variable result in unoptimized code?

- The most powerful optimization for compliler is register allocation. That is it operates the variable from register, than memory.

- Generally local variables are allocated in registers. However if we take the address of a local variable, compiler will not allocate the variable to register.

17. **How does global variables result in unoptimized code?**
    For the same reason as above, compiler will never put the global variable into register. So its bad.

18. **So how to overcome this problem?**
    A: When it is necessary to take the address of variables, (for example if they are passed as a reference parameter to a function). Make a copy of the variable, and pass the address of that copy.

19. **Which is better a char, short or int type for optimization?**

- Where possible, it is best to avoid using char and short as local variables. For the types char and short the compiler needs to reduce the size of the local variable to 8 or 16 bits after each assignment. This is called sign-extending for signed variables and zeroextending for unsigned variables.
- It is implemented by shifting the register left by 24 or 16 bits, followed by a signed or unsigned shift right by the same amount, taking two instructions (zero-extension of an unsigned char takes one instruction).
- These shifts can be avoided by using int and unsigned int for local variables. This is particularly important for calculations which first load data into local variables and then process the data inside the local variables.
- Even if data is input and output as 8- or 16-bit quantities, it is worth considering processing them as 32-bit quantities.

20. **How to reduce function call overhead in ARM based systems?**

- Try to ensure that small functions take four or fewer arguments. These will not use the stack for argument passing. It will copied into registers.

- If a function needs more than four arguments, try to ensure that it does a significant amount of work, so that the cost of passing the stacked arguments is outweighed.
- Pass pointers to structures instead of passing the structure itself.
- Put related arguments in a structure, and pass a pointer to the structure to functions. This will reduce the number of parameters and increase readability.
- Minimize the number of long long parameters, as these take two argument words.
- This also applies to doubles if software floating-point is enabled.
- Avoid functions with a parameter that is passed partially in a register and partially on the stack (split-argument). This is not handled efficiently by the current compilers: all register arguments are pushed on the stack.
- Avoid functions with a variable number of parameters. Varargs functions

21. What is a pure function in ARM terminology?

Pure functions are those which return a result which depends only on their arguments.
They can be thought of as mathematical functions: they always return the same result if the arguments are the same. To tell the compiler that a function is pure, use the special declaration keyword __pure.

```
__pure int square(int x)
{ return x * x;
}
```

Compiler does optimization for pure functions. For example, the values which are allocated to memory can be safely cached in registers, instead of being written to memory before a call and reloaded afterwards.

22. What are inline functions?

- The ARM compilers support inline functions with the keyword __inline.
- This results in each call to an inline function being substituted by its body, instead of a normal call.
- This results in faster code, but it adversely affects code size, particularly if the inline function is large and used often.