EE2016 Microprocessors Theory and Lab

Tutorial 6 (6th week of Aug-Nov 2019 Semester)

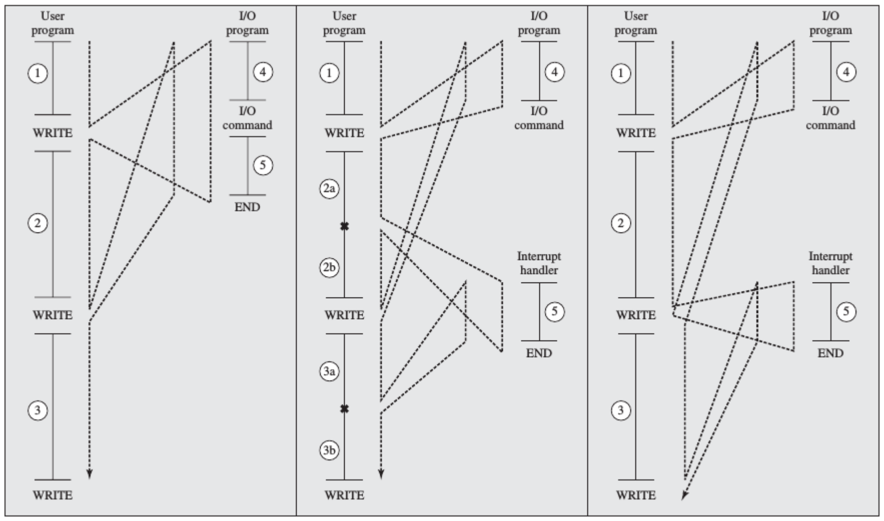
(You may refer AVR manuals & Mazidi)

1 Fill in the blanks

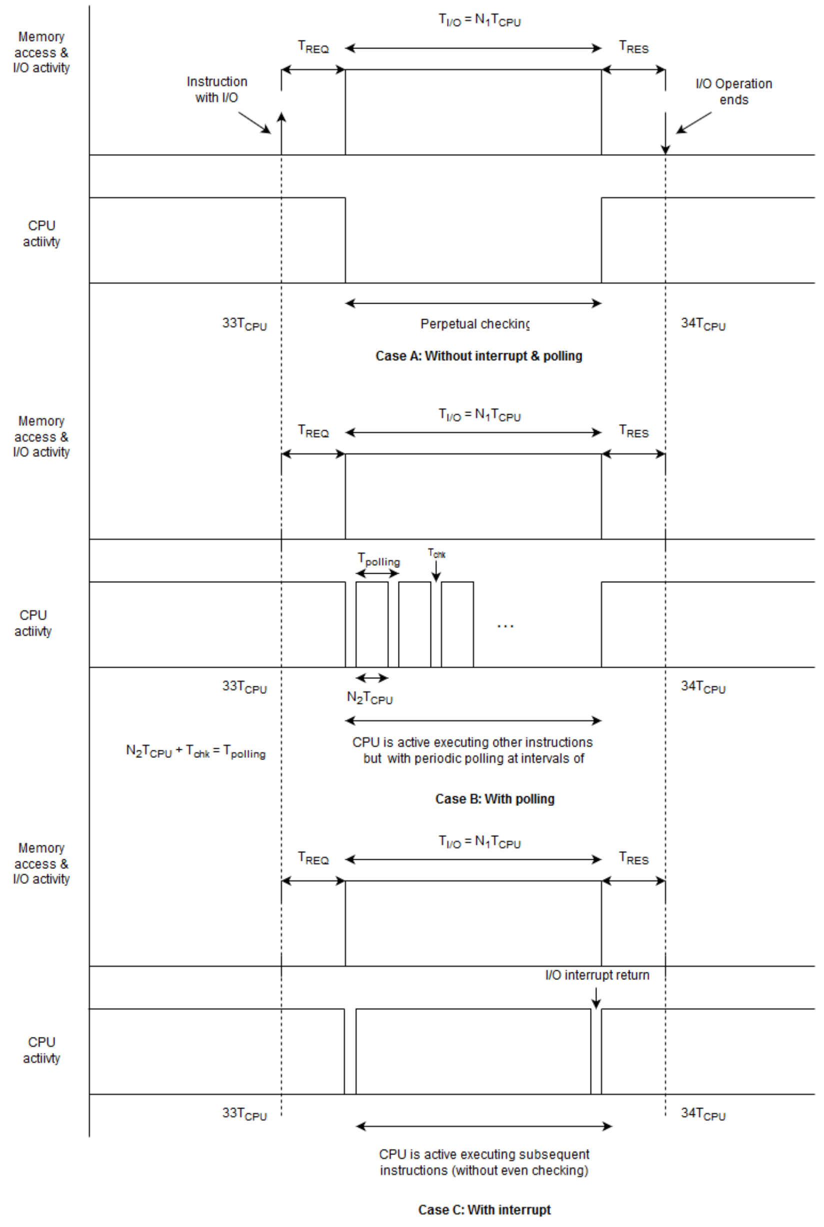
1. Prioritizing the service for such requests from various I/O devices, in polling is .... (possible/ impossible) while in interrupts it is .... (possible/ impossible).
2. The most important advantage of interrupt over polling is ....
3. The difference between RET and RETI is (are) ....
4. The entries in the interrupt vector table give ....
5. The role of Stack pointer in interrupt is (are) ....
6. In AVR, the global interrupt enabling (or disabling) could be done by setting .... bit of the .... register.
7. The instruction (in AVR assembly) to implement global interrupt enabling is .... and for disabling ....
8. Upon power-on reset of the ATmega32, what memory area is assigned to the interrupt vector table? Can programmer change the memory space assigned to the table?

2 Solve all the problems

1. **Polling versus Interrupt versus DMA:** This problem allows one to understand the importance of interrupt / DMA in microprocessors. Though we saw in one of the classes, that 'mov' operation is most common, there are instructions which take much longer time interval as compared to the simple MOV instruction. The operation of printing is one good example, which involves data transfer from memory & I/O (printer). In this problem, we consider four designs of CPU actions in which the data transfer could happen between memory & I/O: (a) CPU waits for the data transfer (without *polling & interrupt*) (b) *polling:* CPU concurrently executes subsequent instructions but polls periodically to see whether I/O operation is over and (c) *interrupt* CPU concurrently executes subsequent instructions, but is interrupted by I/O (that it is done). Refer figure in Stallings as given below



A more convincing comparison picture is given below



Following are the three cases in which a processor could be designed.

(a) *Case A (Without interrupt & polling or DMA):* In this case, after initiating the memory I/O operations in T_{REQ}, the processor waits for the memory I/O operations to complete. During this waiting period, the microprocessor perpetually checks whether the I/O operation is complete or not. Each such checking takes a time interval of T_{chk}. If the I/O job $T_{I/O}$ is over, the time T_{RES} is taken by the I/O module for writing the result of the peripheral operation and the CPU reading it to conclude the I/O operation.

(b) *Case B (With polling):* In this case, the processor is allowed to do other jobs (after initiating the memory I/O operations in T_{REQ}), but in this scheme, the processor polls, at regular intervals $T_{poll,cy} = N_2T_{INS} + T_{chk}$ to check whether the I/O job is over or not. The checking time interval is T_{chk}.

(c) *Case C (With Interrupt)* In this case also, the processor is allowed to do subsequent instructions (concurrently with I/O operations, after initiating the memory I/O operations in T_{REQ}), but is allowed to do other jobs and nothing else. If the I/O job $T_{I/O}$ is over, the processor would be interrupted to know the status in time T_{RES} .

(d) *Case D (With DMA)* In this case, the DMA takes the control of buses from the processor and time required for 'seizing' is $\frac{T_{REQ}}{30}$, while the time for relinquishing the bus control back to the CPU is, $\frac{T_{RES}}{30}$. [Not shown in figure].

Given the following assumptions,

(a) For normal instructions (without I/O) the time required for decode and execute instructions is denoted by $T_{INS} = 6T_{clk}$.

(b) There are a total of 100 such instructions (each of which takes T_{CPU}) out of which the 33rd instruction (& only one such) involves I/O operation.

(c) Execution time for I/O operations is composed of three components (a) preparation for I/O operation which involves CPU time T_{REQ} (b) actual I/O (eg. print job) operation time being $T_{I/O} = N_1 T_{INS}$ (c) response by the I/O peripheral in setting the flag in PIC (whether the operation is success or failure) taking time T_{RES}.

(d) The preparation for I/O operation T_{REQ} means protocol (hand shaking) control information apart from any memory transfer from CPU (peripheral being memory module, image printing in a color printer, video card). The memory transfer happens between the data register of the PIC and the CPU or memory. Similarly, the time interval T_{RES} accounts for again the protocol (hand shaking) control overhead for concluding the I/O transaction. This might also involve block memory transfer from the peripheral to the data registers of the PIC (examples: PICs corresponding to scanner, camera, memory module, sensor). For a color image printer $T_{REQ} \gg T_{RES}$, for camera, sensors, scanners, $T_{REQ} \ll T_{RES}$. In this problem, for cases A through C, take $T_{REQ} = 10T_{INS}$ and $T_{RES} = \frac{T_{INS}}{30}$. Case D is an exception as DMA takes care of everything $T_{REQ} = T_{RES} = \frac{T_{INS}}{3}$.

(e) Primitive checking loop (degenerate form of polling) time is T_{chck} where $T_{chck} = \frac{T_{INS}}{3}$.

(f) The result of the execution of instruction at 33, is used up in instruction K (and the processor would not be able to proceed further without executing instruction K).

Note: In figure above, read T_{CPU} as T_{INS}.

Compute

(a) Total clock cycles required in Cases A through D (for the whole program to complete).

(b) Which of the schemes (A through D) above is most efficient, in terms of number of instructions executed in shortest time). Support your arguments with quantitative results.

(c) Compute the value (or range of values) of N_1 and N_2 such that the CPU need not wait for results to execute statement number K.