EE2016 Microprocessors Theory and Lab, Aug - Nov 2024

Tutorial 4 (Classes on 9, 10, 17 & 20th of September, 2024)

Dr. R. Manivasakan, EE Dept, IIT Madras.

Portions: Interfacing CPU with I/O or memory or any peripherals, memory mapped I/O, port mapped I/O, memory mapping in detail, peripheral interfacing, extending internal bus to system buses to connect to memory, extending to other peripherals using PIC registers, keyboard example, key debouncing problem: algorithm, KBD PIC, polling in KBD, motivation to interrupt, interrupt flow charts, process flow and DMA, DMA algorithm, block transfer and cycle stealing.

1 Fill in the blanks

1. The two main types of addressing a peripheral (given a set of peripherals, picking a desired one) are MMIO and PMIO
2. In memory mapped I/O the address space is SHARED (shared by / separate for) memory and PIC registers of peripherals.
3. Example of port mapped I/O based processors is intel x86 microprocessor.
4. Three important registers in PIC of a controller are status, control and data
5. In a byte addressible memory chip of size 256 bytes, the word length is 8, the address bus width is 8, data bus width is 8 and the control pins are RD, WR, chip enable and clk
6. The status signal (in the context of PIC) is ..... (unidirectional / bidirectional) and if it is unidirectional, then its direction is from pic to processor
7. For a keyboard the system data bus connecting to CPU is ..... (unidirectional / bidirectional) and if it is unidirectional, then its direction is from kbd to processor
8. In polling method of handshake, the proactive component is CPU (CPU / Peripheral)
9. Advantage of interrupt over polling is more efficient instead of wasting time
10. Prioritizing the service by microprocessor, for such service requests from various I/O devices, in polling, is ..... (possible/ impossible), while in interrupts it is ..... (possible/ impossible).
11. The criteria for concluding that "a key has been pressed", (in the polling method of solution to the debouncing problem) is 13 number of samples of '0' (no contact) followed by 7 number of samples of '1' (contact).
12. PC value abruptly jumps during instructions: ..... (select from following list: LDI, JMP, subroutine, ADD, interrupt).
13. The role of stack pointer in interrupt is (are) store local value
14. Stack is a FILO, LIFO (FIFO or LIFO or FILO or LILO) system.
15. The number of POP and PUSH operations in a stack, for a specific interrupt are = (equal / necessarily unequal).
16. The stack pointer, points to ..... (variable bottom most location / variable midlocation / fixed top most location) of active area of stack, which uses the convention of PUSHing from top.
17. The meaning of the KIN bit in KBD_STATUS being '1' is that the data (ASCII code) in KBD_DATA is ..... (valid / invalid - valid means the CPU would READ, invalid means CPU has already READ and should NOT READ again).

18. The main difference between a subroutine and an interrupt-service routine is ..... (Hint: in terms of their arrival time?) during execution of a program.

19. Direct Memory Access (DMA) implements large memory transfers independent of processor using the specialised hardware called ..... This is implemented by forcing the CPU to relinquish .....

20. For smaller amount of memory transfer, preferred handshake is ..... (interrupt / DMA), while for larger amount of memory transfer, preferred handshake is ..... (interrupt / DMA).

21. The memory transfer speed in DMA is limited by the ..... (technology of memory & peripheral / CPU speed).

22. During DMA transfer, the CPU can be busy executing instructions that can only use ..... (internal bus / system bus).

2 Match the following

	A		B
1	Memory mapped I/O	a	CPU is proactive
2	Polling	b	status register
3	part of PIC	c	CPU relinquishes system bus
4	disadvantage of polling	d	common address space
5	DMA	e	stores PC value, just before entering subroutine or interrupt
6	interrupt	f	unwarranted constant engagement of CPU
7	Stack (memory)	g	peripheral initiates the communication

3 Compute the following

1. **MMIO in AVR:** The following is the memory map of GPRs, all I/Os (including timers, EEPROM, interrupts, ports, ADC / DACs and other peripherals), internal SRAM and external SRAMs.

S. No	Purpose	Capacity	Address Range	Comments
1	GPRs	32 B	0x0000 to 0x001F	
2	I/O	64 B	0x0020 to 0x005F	
3	Extended I/O	160 B	0x0060 to 0x00FF	
4	internal SRAM	??	0x0100 to 0x10FF	
5	external SRAM	??	0x1100 to 0xFFFF	

   The external SRAM is implemented by a chip with Chip Enable (CE) pin. Evolve a combinational circuit (give the logical expression) which chooses (active high logic) the external SRAM, when the address is within the range 0x1100 to 0xFFFF. [One student suggested that the logical expression is $Y = (A_{15} + A_{14} + A_{13} + A_{12}) \,.\, (A_{11} + A_{10} + A_9 + A_8)$, just by visual inspection of the range. Is he/she correct? Justify]

2. **MMIO versus PMIO (Optional):** Given a processor with address bus of width 35 bits and peripherals as given below, design a memory mapped I/O and a port mapped I/O. Compare the performance, where the performance measure is fraction of available address range, put to good use. Evaluate the same in both the schemes. Peripherals with their PIC memory requirements are given below in the table. The following table could also be used to indicate the address range in memory mapped I/O.

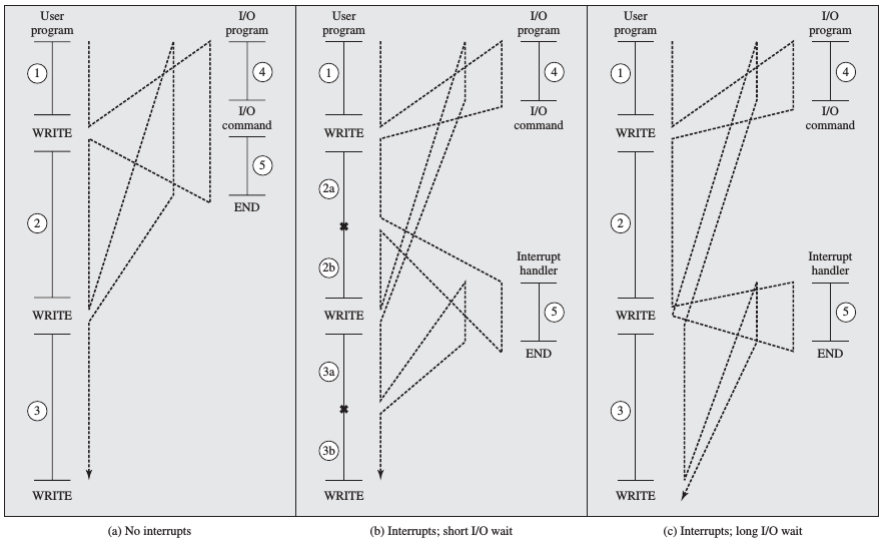
S. No	Type	Capacity	Address Range	Comments
1	HDD	30 GB		
2	RAM	1 GB		
3	ROM	768 MB		
4	Monitor	192 MB		
5	Printer	16 MB		
6	Camera	23 MB		
7	KBD	256 KB		
8	Mouse	256 KB		
9	Headphone	512 KB		
	Total	32 GB		

Now, indicate the distribution of address range for memories in a port mapped I/O along with the corresponding control signals. Note that 1 KB = 1024 Bytes, 1 MB = (1024)x(1024) Bytes & so on. Give a block diagram if necessary. [You need not show the combinational circuits generating the Chip Select (CS) signals].

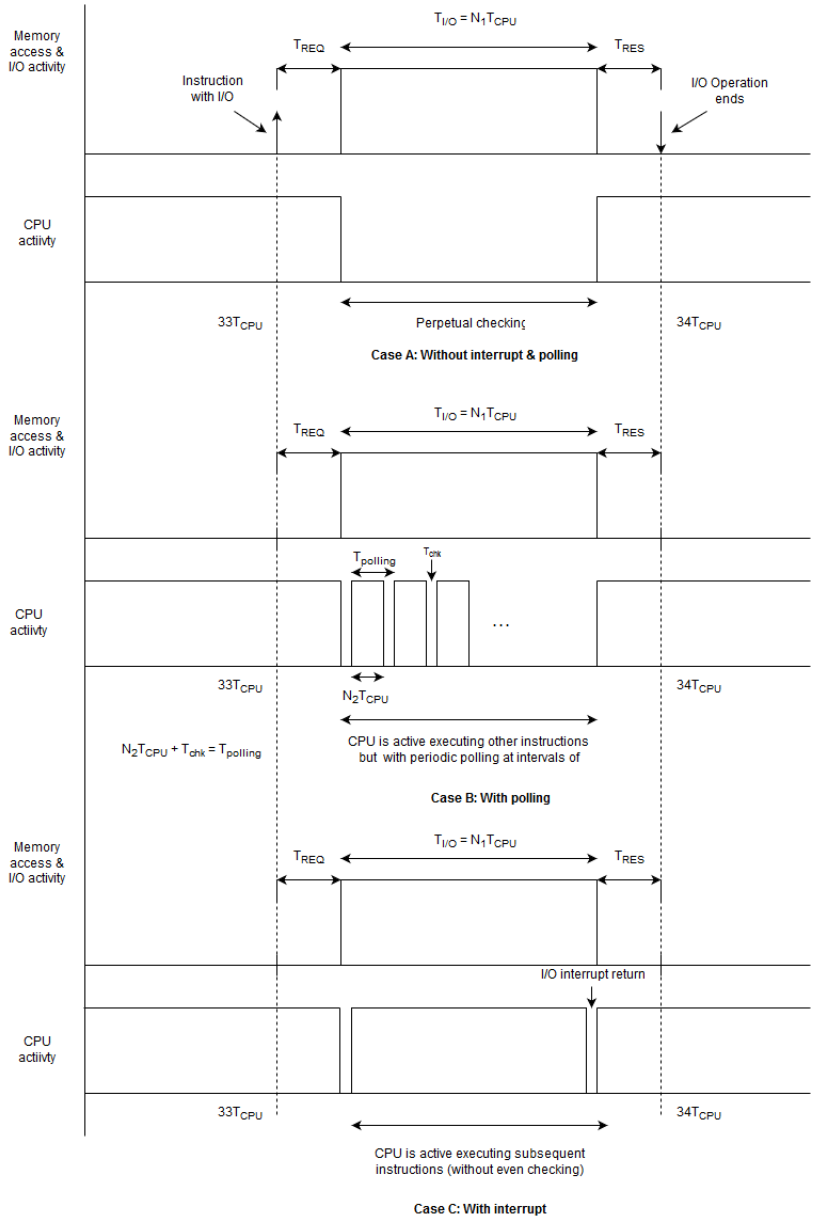
3. **MMIO Design:** Design a memory mapped I/O with the following requirements: A CPU has to be interfaced with the following memory chips and peripherals: Memory chips M0, M1 and M2 each with 512, 1024 and 2048 bytes respectively and peripherals P1, P2 and P3 with PICs of 16, 32 and 64 bytes respectively. Given that these chips are byte addressible, show any sequential circuits you included in your design. [Each chip has CS, RD, WR, address bus and data bus connected]. P1 is a peripheral in which the status register is of a byte, control register is of another byte and rest of the bytes (14 bytes) form data register. P2 is another peripheral in which control and status registers each a byte and the rest (30 bytes) forms a data register memory. Similarly for P3. Show the addressing scheme to the level of registers in PIC corresponding to each peripheral.

4. **Keyboard interfacing:** Given that the keyboard is used by pressing 10 keys consecutively per minute, considering that typing is observed for three minutes, use the flow charts given in class (refer presentation file, for flow charts corresponding to polling and interrupt versions) to compute the total time spent by CPU in each cases (of polling & interrupt versions).

5. **Polling versus Interrupt:** 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, and one of the reasons why it takes less time is because it doesnt involve a peripheral. There are instructions which take much longer time interval as compared to the simple MOV instruction, esp, those that involve I/O. 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 completed and (c) *interrupt* CPU concurrently executes subsequent instructions, but is interrupted by I/O (whenever it needs service or during service, whenever 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):* 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 $T_{poll,cy} = N_2 T_{INS} + T_{chk}$ 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 by the concerned peripheral. 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.

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 $T_{I/O} = N_1 T_{INS} \quad N_1 > 1$ (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 time for preparation for I/O operation T_{REQ} means protocol (hand shaking) control information initially between CPU and the peripheral. For the peripheral being output devices (like memory module, image printing in a color printer, video card etc), T_{REQ} means for eg. the digital data to be printed through the peripheral printer and the parameters of the print command. For the peripheral being input device (eg, camera, sensors, scanners, etc.,) T_{REQ} may mean initial control information from device, like the type of service requested, amount of memory space required, etc.

(e) In the actual I/O operation time, the memory transfer happens between the data register of the PIC and the CPU main memory. For input devices like camera, sensors, scanners, etc., it is the actual data transfer of enduser data from PIC to CPU. For scanner, physical scanning and data being available in its PIC. For output devices, e.g. printer, it is the physical printing time.

(f) The time interval T_{RES} accounts for again the protocol (hand shaking) control overhead for concluding the I/O transaction. For a color image printer $T_{REQ} \gg T_{RES}$. In this problem, for cases A through C, take $T_{REQ} = 10T_{INS}$ and $T_{RES} = \frac{T_{INS}}{30}$.

(g) Primitive checking loop (degenerate form of polling) time is T_{chck} where $T_{chck} = \frac{T_{INS}}{3} = 2T_{clk}$.

(h) The result of the execution of instruction at 33, is used up in instruction $K > 34$ (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 C (for the whole program to complete).

(b) Which of the schemes (A through C) 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.

6. **Memory Access Frequency:** In a 1 Ghz processor, all memory access operations are completed in one clock cycle, in which Load and Store instructions constitute 20 percent of the dynamic instruction count in a program. What is the frequency of memory access operations in the above processor? Assumptions and definitions are given below

   (a) Definition: The dynamic count is the number of instruction executions, including the effect of program loops, which may cause some instructions to be executed more than once.

   (b) Assume that all instructions (including LSI) are executed in 5 clock cycles.

   (c) All memory access (corresponding to either Load or Store) is completed in 1 clock cycle.

   (d) Note, all instructions (in LSA whether it is a memory operation - Load / Store, or an ALU operation) have to be fetched (memory access of 1 clock cycle)

7. **Priority Among Interrupts:** A processor has seven interrupt-request lines, INTR1 to INTR7. Line INTR7 has the highest priority and INTR1 the lowest priority. Design a priority encoder circuit (show truth table, Karnaugh's map & expression for output(s)) that generates a 3-bit code representing the request with the highest priority. See https://en.wikipedia.org/wiki/Priority_encoder.