EE2016 Microprocessors Theory and Lab, Aug - Nov 2024.

Tutorial 5 Solutions (Classes on 14th, 15th, 21st & 22nd of October, 2024)

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

Portions: Keyboard interfacing, performance assessment: CPI till MIPS, Amdhal's law, Gustafson's law, memory management: regular mapping, combination circuit feeding CS, irregular mapping.

1 Fill up the blanks:

1. The purpose of KIN bit in the KBD_STATUS of keyboard PIC is to tell the microprocessor that a specific key has been pressed and the corresponding ASCII code os available in KBD_DATA register
2. The KBD_STATUS register can only be written into (read / written into) by keyboard peripheral.
3. The KIN bit in KBD_STATUS register is used for polling (polling / interrupt / both polling & interrupt schemes) method of handshake.
4. Order the rates in ascending order: (a) rate of keys pressed (b) rate of CPU cycles (c) rate at which the display device could display characters: (1) rate of keys pressed (2) rate at which the display device could display characters and (3) rate of CPU cycles
5. The MOV operation in AVR processor is accomplished in 1 (1/2/3) clock cycle(s).
6. Bus and clock cycles are same (same / different).
7. Memory cycle time is different from (same as / different from) memory access time.
8. CPU cycle is the time corresponding to CPU executing the decode and execute (decode / execute / memory referencing / decode and execute) cycles. It doesnt include the time corresponding to memory referencing (decode / execute / memory referencing / decode and execute) cycles.
9. The meaning of "time to market" (in the context of factors which decide the choice of muC) is time for design, development, prototype testing and introducing a muC based product in the market.
10. The cycles referred in NCPI is clock (instruction / clock / CPU / memory) cycle.
11. Billion (Giga) Instructions per Second (GIPS) is given by (in terms of clock frequency of the CPU and NCPIav) $\frac{f}{(NCPI_{av})\times 10^9}$ where f is the clock frequencies in Hz.
12. Given a task, TNI (total number of instructions) corresponding to that task in a CISC is less than (less than / equal / greater than) the corresponding TNI for RISC. Reason(s) is (are) (a) in a CISC the instructions are complex and does more than one basic operations, (b) in a RISC the instructions form minimal set and are non-overlapping
13. The reason(s) that the cache and memory hierarchy affects $a = \frac{N_{mmry}}{N_{CPU}}$ is (are) cache affects the N_{mmry} and thus also the ratio a
14. Given a task, TNI in the corresponding assembly program, does not depend (depends / does not depend) on process technology (used to fabricate the muP / muC).

15. Given a task, TNI in the corresponding assembly program, `does not depend` (depends / does not depend) on cache and memory hierarchy.

16. Given that p, being the number of CPU operations for decode & execute part of a given instruction, m being the number of memory cycles for memory access operations corresponding to a given instruction, N_{CPU} being the number of clock cycles per CPU operation, N_{mmry} being the number of clock cycles per memory operation, $a = \frac{N_{mmry}}{N_{CPU}}$, then the compiler affects `TNI, p and m` (choose from the list TNI (Total Number of Instructions), p, m, a and T_{clk} or combination thereof).

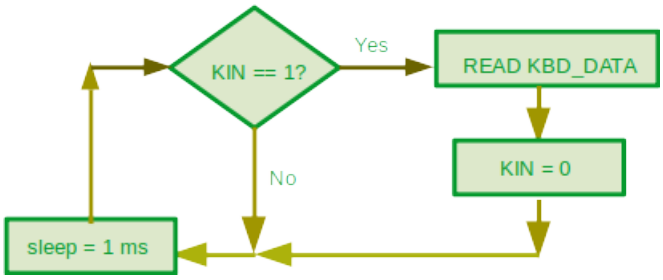
17. Matrices exhibit, inherently the parallelism in computations of `their additions` (their additions / their determinants / their eigen values).

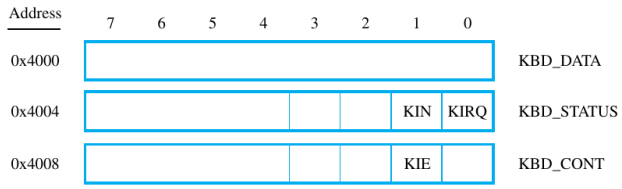
18. The assumption(s) in Amdahl's law in parallel computing is (are) `(1) the running time of the serial part of the program is independent of the number of processors (2) the entire problem is of fixed size for work load so that the total amount of work to be done in parallel is also independent of the number of processors.`

19. Given muC with address bus width (n) being 16 bits ($n = 16$) and data bus width being 8 bits ($m = 8$), interfaced with a memory chip (whose all locations are accessible with the above address bus), has the total memory 2^{16} bytes.

2 Solve ALL the problems

1. **Keyboard Interfacing in AVR:** Write the AVR code for the keyboard interfacing problem corresponding to the flow chart given below. Use the polling method





Solutions:

Refer Hamachar pp 98 - 104 for background information.

The following is the AVR assembly program written by TAs. I have not verified. Report bugs if you find them.

.....

Keyboard_Modified.asm ; Contributed by niyas & Amal;

; Replace with your application code

; Read from the address 4004 2nd bit from LSB. If that bit = 1, means a key is pressed. Otherwise wait for 1 ms.

START: LDI ZH,0x40;

LDI ZL,0x04;

```
REPEAT:     LPM R0,Z; % Read from address 4004 to the register r0
            LDI R16,0x02;
            AND R0,R16;
            BREQ REPEAT ; Not equal means a key is pressed, hence read the key from the location
4000
            LDI ZH,0x40;
            LDI ZL,0x00; ; Setting the location where the pressed key is going to come
            LPM R2, Z ; Read from that location ;
                              ;Create a sleep time for 1 ms 50 X 250 = 12500
                              ;cycles. 12500 X 0.083 = 1 ms
            LDI R17, 0x32
CONTINUE2: LDI R18, 0x0FA
CONTINUE1: DEC R18
            BRNE CONTINUE1
            DEC R17
            BRNE CONTINUE2
            RJMP START
;..... End of the program .....
```

Instructions for emulating the program in PIC studio 7: Use stopovers. Need to manually edit the program flash directly, as in the following:

(a) Change the value at 0x4004 (KIN value).

(b) Change the value at 0x4000 (enter the ASCII value of the keypressed).

2. **DMA:** Transfer of bus control in either direction, from processor to I/O device or vice versa, takes 250 ns. One of the I/O devices has a data transfer rate of 50 KB/s and employs DMA. Data are transferred one byte at a time. Suppose we employ DMA in a burst mode. That is, the DMA interface gains bus mastership prior to the start of a block transfer and maintains control of the bus until the whole block is transferred. For how long would the device tie up the bus when transferring a block of 128 bytes?

   Solutions:

   Time to transfer a block of 128 bytes $T_{b,128} = 250 \times 10^{-9} \times 2 + \frac{128}{(50 \times 1024)} = 0.5\mu Sec + \frac{1}{400} sec = 2.50005 mSec$

3. **Performance Assessment in Microprocessor (0):** State and prove both Amdahl's law and Gustafson's law.

   (a) Amdahl's law

   (b) Gustafson's law

   Solutions:

   (a) Refer to class notes / ppt / wiki

   (b) Refer to class notes / ppt / wiki

4. **Performance Assessment in Microprocessor (1):** Consider the execution of a program which results in the execution of 2 million instructions on a 400-MHz processor. The program consists of four major types of instructions. The instruction mix and the CPI for each instruction type are given below based on the result of a program trace experiment:

Instruction Type	CPI	Instruction Mix
Arithmetic & logic	1	60 %
Load/ store with cache hit	2	18 %
Branch	4	12 %
Memory reference with cache miss	8	10 %

(a) Compute the average CPI when the program is executed on a uniprocessor with the above trace results.

(b) Compute the corresponding MIPS rate.

Solutions:

(a) Average CPI $= 0.6 \times 1 + 0.18 \times 2 + 0.12 \times 4 + 0.1 \times 8 = 2.24$ average number of cycles per instruction

(b) MIPS rate $= \frac{400 \times 10^6}{2.24 \times 10^6} \approx 178\ s^{-1}$

5. **Performance Assessment in Microprocessor (2):** Consider problem 3, for the calculation of average CPI and MIPS rate. Now, assume that the program can be executed in eight parallel tasks or threads with roughly equal number of instructions executed in each task. Execution is on an 8-core system with each core (processor) having the same performance as the single processor originally used. Coordination and synchronization between the parts adds an extra 25,000 instruction executions to each task. Assume the same instruction mix as in the example for each task, but the CPI for memory reference with cache miss is increased to 12 cycles due to contention for memory.

   (a) Determine the average CPI.

   (b) Determine the corresponding MIPS rate.

   (c) Calculate the speedup factor for fixed work load (execution time varies).

   (d) Compare the actual speedup factor with the theoretical speedup factor determined by Amdhal's law.

   (e) Assume that the workload increases linearly with the number of parallel processors available. In that case, calculate the speedup factor (execution workload increases, while execution time is constant).

   (f) Compare the actual speedup factor with the theoretical speedup factor determined by Gustafson's law.

   Solutions:

   (a)

Instruction Type	CPI	Instruction Mix
Arithmetic & logic	1	60 %
Load/ store with cache hit	2	18 %
Branch	4	12 %
Memory reference with cache miss	12	10 %

   Average CPI $= 0.6 \times 1 + 0.18 \times 2 + 0.12 \times 4 + 0.1 \times 12 = 2.64$. The CPI has increased due to the increased time for memory access due to multi-core processors competing for memory.

   (b) $MIPS_{parallel} = \frac{400}{2.64} = 152$. There is a corresponding drop in the MIPS rate in parallel processing case.

   (c) Speed up due to multi-core processors is defined as the ratio of time to execute program on a single processor T_s to the time T_8 to execute program on $N = 8$ parallel processors. $T_s = \frac{TNI}{(MIPS \times 10^6)} = \frac{2 \times 10^6}{178 \times 10^6} = 11\ ms$. To compute the corresponding time for the 8 processor case, we observe that each processor executes one-eigth of the 2 million instructions plus 25,000 overhead instructions.

Total instructions each processor has to execute $TNI_8 = \frac{2\times10^6}{8} + 0.025 \times 10^6$ instructions. For this case, the execution time for each of the 8 parallel processors $T_8 = \frac{TNI_8}{MIPS_{parallel}} = 1.8 \quad ms.$ Therefore, we have speedup $S = \frac{T_s}{T_8} = \frac{11}{1.8} = 6.11$

(d) The answer to this question depends on how we interpret Amdahl's law. Observe that there are two problems in the parallel system: (1) overhead instructions due to parallelisation to coordinate between threads, (2) there is a contention for memory access. The way that the problem is stated, none of the code is inherently serial. All of it is parallelizable, but with scheduling overhead. One could argue that the memory access conflict means that to some extent memory reference instructions are not parallelizable. If we ignore this fact, then $f = 1$. Then, Amdahl's law reduces to speedup $S_{Th} = N = 8$ for this case. Thus, the actual speedup is only about 75% of the theoretical speedup.

(e) In this case T (for executing 2 million instructions in single processor case & for executing more number of instructions in multicore processor case) is fixed. Hence, $T = T_S = \frac{2\times10^6}{178\times10^6} = \frac{2}{178} = 11 \quad ms.$ $T = T_{prlll} = T_{s,8} + T_{s,25000}$ where $T_{s,8}$ is the time required by a single processor in the parallel scheme to execute the $\left(W_{s,T_{s,8}} = \frac{W_{prlll}}{8}\right)$ instructions and $T_{s,25000} = \frac{25000}{178\times10^6}$ is the time required to execute the overhead instructions which are not parallizable (hence each processor has to execute as in Amdahl's law). Hence, $T_{s,8} = \frac{2}{178} - \frac{0.025}{178}$ and $W_{prlll} = 8W_{s,T_{s,8}} = 8 \times MIPS_s \times T_{s,8} = 8 \times 178 \times 10^6 \left(\frac{2}{178} - \frac{0.025}{178}\right)$. Speedup factor is given by $S = \frac{W_{prlll}}{W_s} = \frac{8\times10^6\times(2-0.025)}{2\times10^6} = 4\,(2 - 0.025) = 7.9.$

(f) Theoretical speedup in latency of the execution of whole task at fixed time T in general for a parallelisation factor of a, $(1 - a) + aN$ where N = 8 (Gustafson's law). Here, a = 1 and hence S=8. This is the theoretical bound while the more practical (due to cache miss due to contention in parallelisation) the figure is 7.9 as above.

6. **Performance Assessment in Microprocessor (3):** A processor accesses main memory with an average access time of T_2. A smaller cache memory is interposed between the processor and main memory. The cache has a significantly faster access time of $T_1 < T_2$. The cache holds, at any time, copies of some main memory words and is designed so that the words more likely to be accessed in the near future are in the cache. Assume that the probability that the next word accessed by the processor is in the cache is H, known as the hit ratio.

   (a) For any single memory access, what is the theoretical speedup of accessing the word in the cache rather than in main memory?

   (b) Let T be the average access time. Express T as a function of T_1, T_2, and H. What is the overall speedup as a function of H?

   (c) In practice, a system may be designed so that the processor must first access the cache to determine if the word is in the cache and, if it is not, then access main memory, so that on a miss (opposite of a hit), memory access time is $T_1 + T_2$. Express T as a function of T_1, T_2 and H. Now calculate the speedup and compare to the result produced in part (b).

Solutions:

(a) Speedup $S = \frac{time\ to\ access\ the\ main\ memory}{time\ to\ access\ in\ cache} = \frac{T_2}{T_1}$

(b) Average access time (simple scheme) $T = T_2\,(1 - H) + HT_1$. Speedup $S = \frac{access\ time\ without\ cache}{time\ to\ access\ with\ cache\ incorporated} = \frac{T_2}{T} = \frac{T_2}{T_2(1-H)+HT_1} = \frac{1}{(1-H)+H\frac{T_1}{T_2}}$

(c) Average access time (practical scheme) $T = (T_1 + T_2)\,(1 - H) + HT_1$. Or $T = T_2\,(1 - H) + T_1$. $S = \frac{access\ time\ without\ cache}{time\ to\ access\ with\ cache\ incorporated} = \frac{T_2}{T} = \frac{T_2}{T_2(1-H)+T_1} = \frac{1}{(1-H)+\frac{T_1}{T_2}}$