CSC 230 Assignment 4
Fall 2015

This assignment has two parts:

Written Part: Complete the Assignment 4 written part by uploading a pdf document to the Assignments link of the Connex site. **Submit it on conneX before 10am on November 30, 2015.**

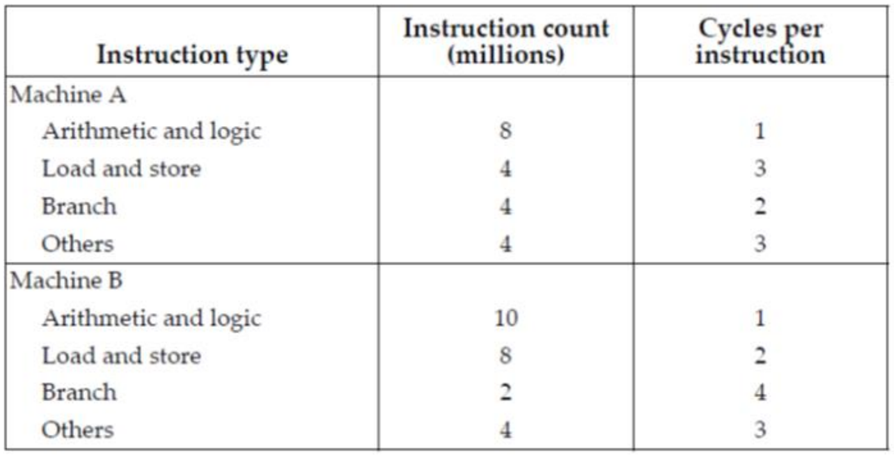
Programming Part: Complete the Assignment 4 programming part by uploading a file (with extension .c) to the Assignments link of the Connex site. **Submit on ConneX before 6pm December 4, 2015**

Please be concise with your answers and if possible highlight your final answer.

Assignment 4: Written Part

Question 1 [10] (Performance)

Consider two different CPUs (machines), with two different instruction sets, and with the same clock rate of 200MHz. The following measurements are recorded on these two machines running a given program:



Instruction type	Instruction count (millions)	Cycles per instruction
Machine A		
Arithmetic and logic	8	1
Load and store	4	3
Branch	4	2
Others	4	3
Machine B		
Arithmetic and logic	10	1
Load and store	8	2
Branch	2	4
Others	4	3

You need to show your calculations to get marks!!

(a) [1] What is the clock period (in *ns*) of the CPUs?

1/200Mhz = 5 ns

(b) [1] What is the execution time (in *ms*) on Machine A for this program?

$(8*1+4*3+4*2+4*3)10^6*5$ns = (200 ms)

(c) [1] What is the execution time (in *ms*) on Machine B for this program?

(230 ms)

(d) [1] What is the performance ratio of Machine A over Machine B (i.e., Performance$_A$/Performance$_B$)? Which machine is better?

230/200 = 1.15; Machine A is 15% better.

(e) [2] Determine the effective CPI for each machine (show your calculations).

Machine A = (((8*1)+(4*3)+(4*2)+(4*3))/(8+4+4+4)) = 2

Machine B = (((10*1)+(8*2)+(2*4)+(4*3))/(10+8+2+4)) = 1.917

(f) [1] If Machine A clock rate is increased by 20%, what will be the speed-up achieved for Machine A?

Original time / (Original time /1.20) = 1.20

(g) [1] If Machine B design is improved so that only 6 M Arithmetic and Logic instructions are required, rather than 10M as shown in the above table, what will be the speed up achieved for Machine B?

The new execution time: **$(6*1+8*2+2*4+4*3)*10^6*5$ns=210 ms**

Speedup = 230/210 = 1.0952

(h) [2] A common measure of performance for a processor is the rate at which instructions are executed, expressed as millions of instructions per second (MIPS), referred to as the MIPS rate. We can express the MIPS rate in terms of the clock rate and CPI as follows:

$$\text{MIPS rate} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

State the MIPS rate for each machine above.

Machine A = 200*10^6 / (2*10^6) = 100 MIPS

Machine B = 200*10^6 / (1.917 * 10^6) = 104 MIPS

Question 2 [10] (Pipeline)

A CPU has a clock rate of 1.25Ghz. Let us assume that the CPU has 4 pipeline stages and processing delay of each stage is exactly one clock cycle period.

a) [1] What is the execution time of each instruction passing through all stages?

Clock period = 0.8 ns; 4 stages = 3.2 ns

b) [1] What will be the instruction throughput (instructions/sec) with sequential (serial) (i.e., no pipeline) execution?

1 instruction per every 3.2 ns = 312.5 MIPS

c) [1] What will be the instruction throughput (instructions/sec) with pipeline operation?

1 instruction per every 0.8 ns = 1.25 * 10^9 insturctions/sec

d) [1] What is the speed-up achieved when 500,000 instructions are processed using the pipeline operation?

Speed up = Ts/Tp = mN/(m+N-1) = 3.99

e) [1] What is the speed-up achieved when 1,000,000 instructions are processed using the pipeline operation?

Approximately 4

f) [1] What will be the maximum speed-up achieved with this pipeline? **4**

g) [4] Given that the branch penalty is 10 cycles, the probably of a branch instruction is 15% and the probability that a branch is taken is 50%, what will be the average CPI for this CPU and what will the execution efficiency?

CPI branch = $1 + bP_bP_t = 1 + 10*0.5*0.15 = 1.75$
Execution Efficiency = $1/1.75 = 0.571$

Question 3 [10] (Cache)

As discussed in the class for memory hierarchy to work correctly, the main memory is mapped on to the cache memory. Some of the mapping techniques are direct, fully associative and set associative. The key design parameters are: sizes of main and cache memory, lines (blocks) in the cache, blocks in the main memory and mapping function. For this question let us assume that the size of the main memory is 16Mbytes, the size of the cache is 16Kbytes and the size of the block is 32 bytes and the CPU is operating at 16MHz.

a) [1] How many address bits (lines) are needed for the main memory?

24 bits (2^{24}) = 16M bytes

b) [1] How many blocks of main memory are there? How many bits are needed to identify these blocks?

$2^{24}/2^{5} = 2^{19}$ blocks; 19 bits

c) [1] How many cache lines (slots, blocks) are there and how many bits are needed to identify these?

16K bytes / 32 bytes = 512 = 2^{9} ; 9 bits

d) [1] How many tag bits are needed if the memory uses direct-mapped cache?

10 bits

Our example system has 16 K = 2^14 bytes and the cache line size is the same as the memory block size (2^5). So, there are 2^14 / 2^5 cache lines = 2^9 lines.

The direct mapping from the memory blocks to the cache lines is done using a modulus operation:

- **Into which cache line will the block be placed? Memory block address modulus 2^9.**
- **What is the tag? Memory block address / 2^9**
- **So, how many tag bits are needed? 19 – 9 = 10**

e) [1] How many tag bits are needed if the memory uses fully-associative cache?

19 bits

f) [1] How many tag bits are needed if the memory uses a two-way (set size of 2) set-associative cache?

11 bits

Each set has 2 cache lines. So, in our example there are 2^9 / 2 = 2^8 sets.
The mapping from the memory blocks to the cache sets is still done using a modulus operation:

- **Into which cache set will the block be placed? Memory block address modulus 2^8.**
- **What is the tag? Memory block address / 2^8**
- **So, how many tag bits are needed? 19 – 8 = 11**

g) [1] How many tag bits are needed if the memory uses a 4-way set-associative cache with a set size of 4?

12 bits
Each set has 4 cache lines. So, in our example there are 2^9 / 4 = 2^7 sets.
The mapping from the memory blocks to the cache sets is still done using a modulus operation:

- **Into which cache set will the block be placed? Memory block address modulus 2^7.**
- **What is the tag? Memory block address / 2^7**
- **So, how many tag bits are needed? 19 – 7 = 12**

h) [1] Let us assume that the cache hit rate is 90%, cache access time is 1 cycle and miss penalty is 16 cycles. What is the average memory access time of the CPU?

1*0.9 + 16*0.1 = 2.5 cycles

i) [1] What is the speed-up achieved with the above parameters?

16/2.5 = 6.4

j) [1] What is the speed-up if cache hits ratio changes to 50%?

1*0.5+ 16*0.5 = 8.5 cycles
Speed up = 16/8.5 = 1.89

Assignment 4: Programming Part 20

- Write a C function, `Fibon(n)`, that calculates and returns the `n`th Fibonacci number, where `n` is an integer. (The series of Fibonacci numbers are defined as follows: $F_0 = 0$, $F_1 = 1$, $F_2 = 1$, $F_3 = 2$, $F_4 = 3$, . . .) The function should pass in an integer parameter, `n`, and return an integer value, the corresponding Fibonacci number.

- Next create and test a loop that calls the function, with values of `n` from `n=0` and continuing until the maximum value that returns a *correct* Fibonacci number. Include in your function documentation an indication of this maximum value. Also, provide an explanation of what causes that failure with larger values.

- Finally, expand your program so that, for each consecutive value of `n`, the program causes the `(n modulus 6)`th LED in the LED strip on our AVR 2560 boards to blink `Fibon(n)` times.

Submit a well-documented C program to Connex before 6 pm on the last day of classes this term.