submitting by Rotem Kashani 209073352

Computer Architecture Exercise 1

Consider the following code fragment for a contemporary PC (all numbers given in decimal):

	Instruction	Action	Clock Cycles
	MOV DWORD PTR [EBP-04],0	; i = 0	1
	MOV DWORD PTR [EBP-12],0	; k = 0	1
	MOV DWORD PTR [EBP-16],0	; l = 0	1
L1:	CMP DWORD PTR [EBP-04],100		2
	JGE L2	; break on i ≥ 100	3
	MOV EAX,[EBP-04]	; EAX ← i	1
	SHL EAX,1	; EAX ← 2 * EAX	1
	MOV [EBP-08],EAX	; j ← EAX	1
	ADD [EBP-12],EAX	; k ← k + EAX	3
	SUB [EBP-16],EAX	; l ← l - EAX	3
	INC DWORD PTR [EBP-04]	; i++	3
	JMP L1	; loop	3
L2:	RET		3

Each register holds a 32-bit integer (a **DWORD** in Intel terminology).

1. Find the number of instruction cycles required to execute the program as written.

   Define N – number of iterations in a loop.
   The total clock cycles is 1+ 1+ 1+ N*(2+3) + (N-1) *(1+ 1+ 1+ 3+ 3+ 3+ 3) + 3 =20*N -9.
   In our case N = 101, because the iteration stops when i = 100 so we get that the number of instruction cycles required to execute the program as written is 20*101-9 =**2011**.

2. Registers EBX, ECX, and EDX are reserved. Variables can be allocated to registers ESI and EDI instead of main memory. Operations on registers require 1 CC instead of 2 or 3.
   Carry out the exchange.
   Find the number of instruction cycles required to execute the updated program.
   Find the relative speedup compared to the original program.

   In lines 1-3 there are no changes in line 2 we decrease the number of cycles from 2 to 1, and in lines 9 and 11 we decrease the number of cycles from 3 to 1.
   So if we define N like in question 1 to be the number of iterations, we will get that the number of the corrected program is:
   The total clock cycles is 1 + 1 + 1 + N*(1 + 3) + (N – 1)*(1 + 1 + 1 + 1 + 3 + 1 + 3) + 3 = 15*N – 5.
   The reference to calculating the three units at the beginning and adding 3 stems from

the same reasons which I explained in questions 1.
For N = 101 we will get that the number of instruction cycles required to execute the updated program is 15*101-5 = **1510**.
The relative speedup defined to be the ratio between CC^{old} to CC^{new} , and therefore we will get: 2011/1510 = **1.331**.

3. The CPU in a contemporary PC includes 8 additional registers R0, R1, ..., R7.
   Use these registers to improve the program.
   Find the number of instruction cycles required to execute the updated program.
   Find the relative speedup compared to the original program.

   We will define the registers in the alternative program as follows:
   [EBP-04] = R0
   [EBP-12] = R1
   [EBP-16] = R2
   [EBP-08] = R3
   We will define again the number of iterations in the loop to be N, and we will get that the number of cycles needed to run the improved program will be:
   The total clock cycles is 1 + 1 + 1 + N*(1 + 3) + (N-1)*(1 + 1 + 1 + 1 + 1 + 1 + 3) + 3 = 13*N – 3.
   The reference to calculating the three units at the beginning and adding 3 stems from the same reasons which I explained in questions 1.
   For N=101 we get that the number of cycles needed to run the revised program will be: 13*101 – 3 = **1310**.
   The relative improvement in run time of the original program: 2011/1310 = **1.535**.

4. The program can be split into two threads with identical code except for the variable i.
   On a dual core processor, each thread runs in parallel (each thread on a separate core, so that both threads begin at the same time). To good approximation, the run time for each thread will be identical and the program finishes when one thread finishes.
   The thread for CPU0 runs for i = 0 and i < 50.
   The thread for CPU1 runs for i = 50 and i < 100.
   Find the number of instruction cycles required to execute the updated program.
   Find the relative speedup compared to the original program.
   The speedup is less than 2, despite the two cores. The clock cycles for the sequential version of question 3 can be split into two groups: those that run fewer times in the threaded version and those that run the same number of times.
   How many clock cycles are in each group?

   The number of cycles needed to run the modified program when each thread in a separate core and they run concurrently: For each thread we will define N=51 and we will get that for each thread the number of cycles needed to run the corrected program

will be:
The total clock cycles is 1+1+1+N*(1+3) + (N-1)*(1+1+1+1+1+1+3) + 3 = 13*N–3.
And we get: 13*51 – 3 = 660, so the number of cycles needed to run the entire program is **660** because the program ends when thread ends.
The relative improvement in relation to the running time in program 3 will be: 1310/660=**1.984**.
When we divide the serial version in question 3 into two groups - those who will run less times in the division and those who will run same times, we will get that in the group that run the same number of times, initialization and termination command lines are included - that is, lines 1-3 and the last line. The number of running cycles of this group will be 1+1+1+3 = 6.
In the group that run less times, in the division into threads, we divide the body of the loop into two threads, therefore the number of running cycles of this group will be 51*4 + 50*9 = **654** per single thread, and 101*4 + 100*9 = **1304** (when the 4 is for testing i and comparing it to 100, and the 9 is for the body of the loop).