CSCI 4717 – Computer Arch.
Fall 2012 Superscalar Exercise

Name: _____

1. The figure to the right represents the execution of 6 instructions on an "in-order-issue/in-order-completion" machine.

   Assume that there are two true data dependencies (write-read) in the sequence, one between I4 and I5, i.e., I5 depends on the completion of I4 before it can execute, and one between I5 and I7. As for resource dependencies, assume that any instructions that share a column in the execute stage require the same resource.

Decode		Execute			Write		Cycle
I1	I2						1
I3	I4	I1	I2				2
I3	I4		I2				3
	I4	I3			I1	I2	4
I5	I6	I4					5
I7	I8		I5	I6	I3	I4	6
I7	I8			I6			7
		I7	I8		I5	I6	8
					I7	I8	9

   Fill in the figure below to show how quickly an "out-of-order-issue/out-of-order-completion" machine could execute these instructions. Be sure to show which instructions are in the window too.

Decode		Window	Execute			Write		Cycle
								1
								2
								3
								4
								5
								6
								7
								8
								9

2. The tables on the following page represent data collected using the high performance counter on a Dell Optiplex 780 with an Intel® Core™2 Duo CPU running at 3.33 GHz with 8.0 GB of RAM. The values represent the elapsed times for a loop to execute 2 to 12 simple additions on 2 to 12 integer values. The number of integer values updated is equivalent to the number of operations, i.e., for a loop that contained four operations, there were four integer variables updated. The page after that contains two graphs, one comparing the elapsed times for no dependencies against those with true dependencies and another showing the incremental differences in time between n-1 operations and n operations. The page after that shows a portion of the code used to generate the data. The last page shows the disassembled code for the two loops (with and without dependencies) that perform three additions on three variables.

   On the D2L quiz titled, "Superscalar In-Class Exercise," write your answers to the following problems.

   a. Examine at the execution time and incremental differences in execution time and identify trends and/or characteristics of the data.

   b. Propose a minimum of two characteristics of the superscalar processor that you think are causing the characteristics you identified in the previous question.

   c. Examine the sample code and discuss whether or not there might be additional factors that might affect these measurements. Feel free to use the assembly language code to support your proposals.

   d. What other information would it have been nice to have to perform your evaluation?

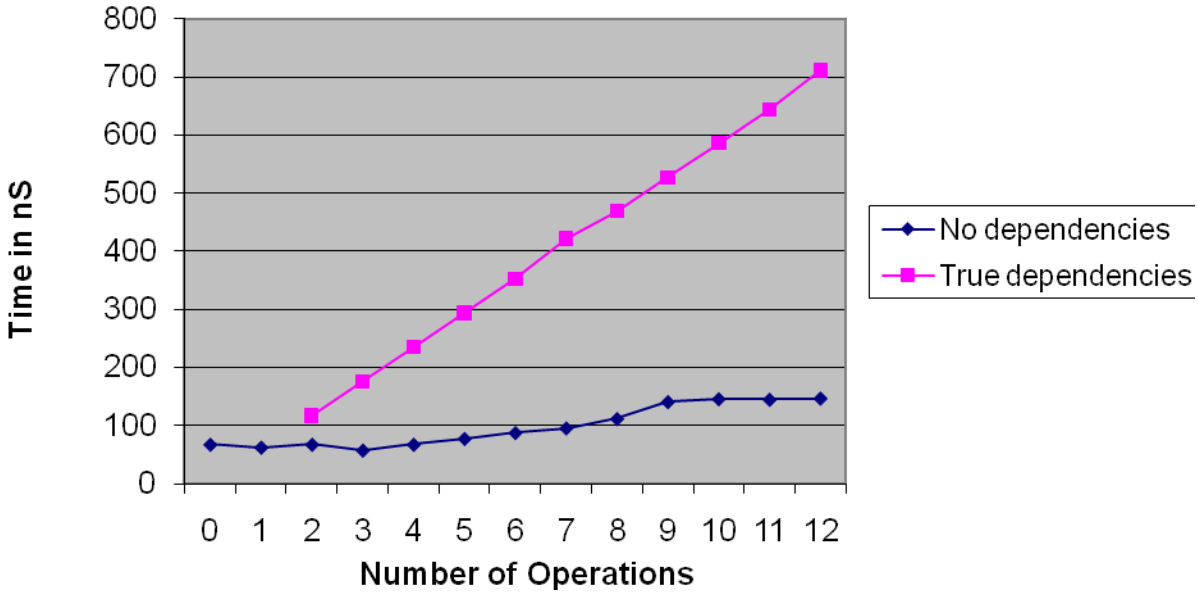
Raw Data Collection

# of ops	Depend-encies	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Sample 6	Sample 7	Sample 8	Sample 9	Sample 10	Average
0	None	68	69	68	69	68	69	69	68	68	68	68
1	None	63	63	63	63	63	63	63	63	63	63	63
2	None	69	68	69	68	69	68	68	69	69	69	68
2	True	117	118	117	118	117	118	118	118	118	117	117
3	None	59	58	59	59	59	58	58	58	58	59	58
3	True	176	176	176	176	176	176	177	177	177	176	176
4	None	69	69	68	68	68	69	68	68	68	68	68
4	True	234	234	235	240	235	234	235	235	234	235	235
5	None	78	79	78	79	78	79	78	78	79	78	78
5	True	295	294	295	294	294	293	294	294	294	294	294
6	None	88	88	88	88	88	88	88	88	88	88	88
6	True	354	351	353	354	354	354	353	352	353	354	353
7	None	96	98	97	96	96	97	97	97	96	95	96
7	True	416	411	431	430	427	412	420	424	423	427	422
8	None	111	110	112	118	110	111	118	110	112	111	112
8	True	469	469	469	469	469	469	469	469	469	469	469
9	None	141	141	142	141	142	141	141	142	142	142	141
9	True	528	528	527	528	527	528	528	527	527	527	527
10	None	146	146	147	147	147	147	147	147	147	147	146
10	True	586	586	586	589	586	586	586	586	586	586	586
11	None	146	146	145	147	145	145	146	146	145	147	145
11	True	645	646	645	644	645	645	645	644	645	644	644
12	None	147	148	148	150	147	147	147	148	147	150	147
12	True	706	703	708	703	734	713	712	706	730	704	711

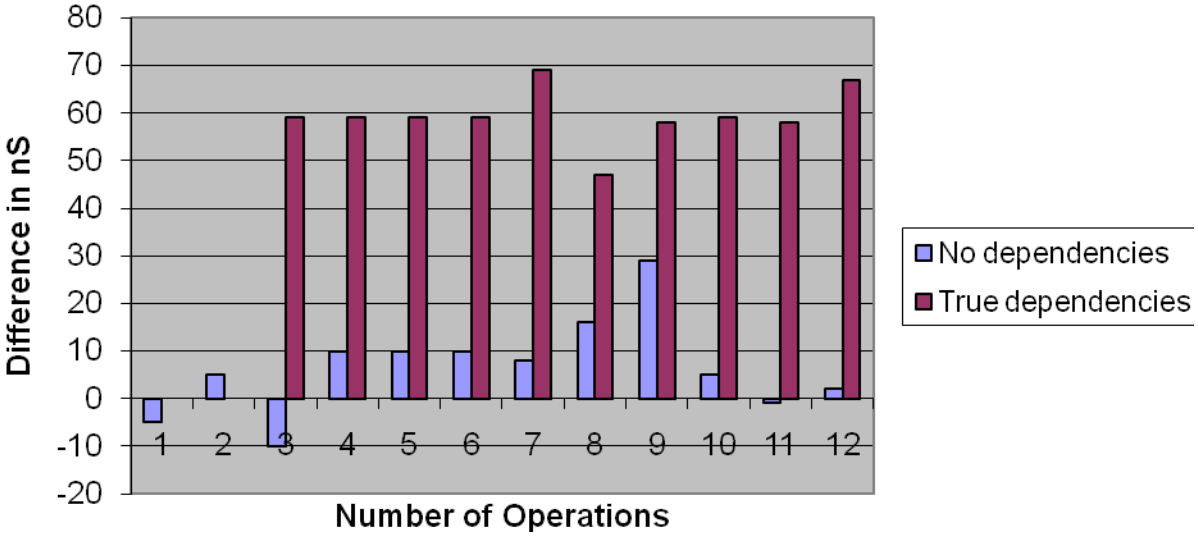
Data Summary

# of ops	wo/dep	difference	w/dep	difference
0	68			
1	63	-5		
2	68	5	117	
3	58	-10	176	59
4	68	10	235	59
5	78	10	294	59
6	88	10	353	59
7	96	8	422	69
8	112	16	469	47
9	141	29	527	58
10	146	5	586	59
11	145	-1	644	58
12	147	2	711	67

Elapsed Execution Time



Incremental Differences in Elapsed Execution Time



Portion of Original C++ Code

```
QueryPerformanceFrequency(&frequency);
QueryPerformanceCounter(&time_reading[0]);
for (i=0; i<10000; i++){
      a += 5;
      b += 20;
}
QueryPerformanceCounter(&time_reading[1]);
for (i=0; i<10000; i++){
      b = a + 5;
      a = b + 20;
}
QueryPerformanceCounter(&time_reading[2]);
...
...   // Large portion of code snipped from this area
...
QueryPerformanceCounter(&time_reading[20]);
for (i=0; i<10000; i++){
      a += 5;
      b += 20;
      c += 3;
      d += 19;
      e += 2;
      f += 24;
      g += 85;
      h += 11;
      j += 23;
      k += 9;
      l += 72;
      m += 54;
}
QueryPerformanceCounter(&time_reading[21]);
for (i=0; i<10000; i++){
      b = a + 5;
      c = b + 20;
      d = c + 3;
      e = d + 19;
      f = e + 2;
      g = f + 24;
      h = g + 85;
      j = h + 11;
      k = j + 23;
      l = k + 9;
      m = l + 72;
      a = m + 54;
}
QueryPerformanceCounter(&time_reading[22]);

cout << "2 integer ops wo/dep = " << time_reading[1].QuadPart -
            time_reading[0].QuadPart)/(frequency.QuadPart/1000000000) < "\n";
cout << "2 integer ops w/dep = " << time_reading[2].QuadPart -
            time_reading[1].QuadPart)/(frequency.QuadPart/1000000000) < "\n";
...
...   // Large portion of code snipped from this area
...
cout << "12 integer ops wo/dep = " << (time_reading[21].QuadPart -
            time_reading[20].QuadPart)/(frequency.QuadPart/1000000000) << "\n";
cout << " 12 integer ops w/dep  = " << (time_reading[22].QuadPart -
            time_reading[21].QuadPart)/(frequency.QuadPart/1000000000) << "\n";
```

Short Section of Disassembled Code

```
      for (int i=0; i<10000; i++)
013716A8  mov         dword ptr [i],0
013716B2  jmp         main+233h (13716C3h)
013716B4  mov         eax,dword ptr [i]
013716BA  add         eax,1
013716BD  mov         dword ptr [i],eax
013716C3  cmp         dword ptr [i],2710h
013716CD  jge         main+25Ch (13716ECh)
      {
          a += 5;
013716CF  mov         eax,dword ptr [ebp-0Ch]
013716D2  add         eax,5
013716D5  mov         dword ptr [ebp-0Ch],eax
          b += 20;
013716D8  mov         eax,dword ptr [ebp-18h]
013716DB  add         eax,14h
013716DE  mov         dword ptr [ebp-18h],eax
          c += 3;
013716E1  mov         eax,dword ptr [ebp-24h]
013716E4  add         eax,3
013716E7  mov         dword ptr [ebp-24h],eax
      }
013716EA  jmp         main+224h (13716B4h)
      QueryPerformanceCounter(&time_reading[5]);
013716EC  lea         eax,[ebp-138h]
013716F2  mov         esi,esp
013716F4  push        eax
013716F5  call        dword ptr [__imp__QueryPerformanceCounter@4 (137C234h)]
013716FB  cmp         esi,esp
013716FD  call        @ILT+390(__RTC_CheckEsp) (137118Bh)
      for (int i=0; i<10000; i++)
01371702  mov         dword ptr [i],0
0137170C  jmp         main+28Dh (137171Dh)
0137170E  mov         eax,dword ptr [i]
01371714  add         eax,1
01371717  mov         dword ptr [i],eax
0137171D  cmp         dword ptr [i],2710h
01371727  jge         main+2B6h (1371746h)
      {
          b = a + 5;
01371729  mov         eax,dword ptr [ebp-0Ch]
0137172C  add         eax,5
0137172F  mov         dword ptr [ebp-18h],eax
          c = b + 20;
01371732  mov         eax,dword ptr [ebp-18h]
01371735  add         eax,14h
01371738  mov         dword ptr [ebp-24h],eax
          a = c + 3;
0137173B  mov         eax,dword ptr [ebp-24h]
0137173E  add         eax,3
01371741  mov         dword ptr [ebp-0Ch],eax
      }
01371744  jmp         main+27Eh (137170Eh)
```