214 HW Chapters 6, 7, and 8
Samuel Jentsch

Chapter 8:

10. What problems does an assembler have to overcome in order to produce complete binary code in one pass over the source file?

An assembler usually builds a symbol table to handle relocatable code defined in the assembly file. A one pass assembler can’t build a symbol table and requires absolute addresses for the variables defined in the assembly file. The code likely is not relocatable and must be run in a specific location each time.

How would code written for a one-pass assembler be different from code written for a two-pass assembler?

Code for one-pass assemblers is usually written as absolute code rather than relocatable code. The code must always be loaded into a particular location in memory. This removes the need for the construction of a symbol table needed for relocatable code. Code written for one-pass assemblers uses compile time binding, instructions and data addresses are bound to physical addresses at compile time.

15. Why is the execution environment of a Java class called a virtual machine?

Because the Java Virtual Machine is the environment where Java classes are executed. The JVM is a virtual machine that provides an interface between Java class files and the operating system, and isn’t actually a real machine. The JVM has private memory addresses and simulates an actual machine. The virtual machine allows for execution of Java files on a variety of actual machine types by providing this virtual interface.

How does this virtual machine compare to a real machine running code written in C

The JVM isn’t as efficient as an actual machine running code written in C. The C compiler generates assembly code specifically for the architecture it is running on, while the class files for the JVM are designed for the JVM rather than the actual hardware. While the code written in C is faster, code written for the JVM is more portable since the JVM provides an interface for the code on a variety of architectures.

18. Java bytecode for access to the local variable array for a class is at most 2 bytes long. 1 byte is used for the opcode, the other indicates the offset into the array.

```
        (* * * * * * * *)(* * * * * * * *) -> (OPCODE = 8 bits)(OFFSET = 8 bits)
```

How many variables can be held in the local variable array?

The instruction allows for an 8 bit offset. 2^8 = 256, so 256 variables can be stored in the array.

What happens when this number is exceeded?

An error occurs stating that the possible number of variables has been exceeded. Most likely an out of memory error of some sort.

Chapter 9:

4. RISC, overlapping register windows with: 10 global, 6 input parameter, 10 local, 6 output parameter. How large is each overlapping register window?

Each register window will contain 22 registers (6 input, 10 local, and 6 output registers). The 10 global registers are shared by the windows and are not part of the individual register sets.

10. RISC Machine with 5 register windows

a)

WINDOW 1					
OUT	IN WINDOW2				
	OUT	IN WINDOW3			
		OUT	IN WINDOW4		
			OUT	IN WINDOW5	
				OUT	IN WINDOW1

The 6th procedure call requires the window 1 to be saved to memory, so 5 procedures can be active without saving a register set to memory.

b)

WINDOW 1						
OUT	IN WINDOW2					
	OUT	IN WINDOW3				
		OUT	IN WINDOW4			
			OUT	IN WINDOW5		
				OUT	IN WINDOW1	
					OUT	IN WINDOW2
						OUT

2 registers must be saved to memory (Window 1 and Window 2).

c) The procedure currently stored in WINDOW2 returns. The procedure most recently saved to memory (the procedure that was stored in WINDOW2 before it was overwritten) is retrieved from memory and stored back in WINDOW2.

d) The procedure in WINDOW2 returned, so the procedure call would occur from the procedure in WINDOW1, this would result in WINDOW2 being stored to memory again. 1 window is stored to memory.

18. <u>Describe and compare VLIW/superscalar w/ respect to instruction-level parallelism.</u>

The superscalar model uses superpipelining in conjunction with superscalar hardware components to provide instruction parallelism. Superpipelining allows instructions to be split into stages that require less than half a clock cycle to execute. These stages are than executed in parallel using special hardware (execution units) according to the superscalar model.

Superscalar architectures rely on hardware (designed to handle dependencies) and the compiler (which handles scheduling).

The VLIW model packs independent instructions into one long instruction designed to tell execution units what to do to allow for parallel processing. This approach is entirely dependent on the compiler rather than hardware and the compiler like the superscalar model. The compiler is responsible for handling dependencies and scheduling for parallel execution of instructions. Instructions are fixed at compile time.

24. a)

i) P0 -> M2
    1A through, 2A cross, 3B through.

ii) P4 -> M4
    1A through, 2B through, 3C through.

iii) P6 -> M3
    1C cross, 2A through, 3B cross.

b) P4 -> M4 could occur at the same time as the other 2 accesses, but P0 -> M2 and P6 -> M3 could not occur at the same time because they require different paths through 2A (through vs cross) and 3B (cross vs through).

c) P2 -> M3 is blocked at stage 2A by the cross required by P0 -> M2.

d) P7 -> M7 would not be blocked by P0 -> M2 (uses separate memory).

26. It seems like it should be associative. The connections between the different stages map directly to other stages. It seems like associative memory would be faster for these types of connections since it is designed to map directly to blocks of memory (the connections in the data flow system can map directly to other blocks in the data flow system using associative memory). This should allow for faster traversal of the data flow system.

33. CISC or RISC

1) R
2) C
3) C
4) R
5) C
6) C
7) R
8) C
9) R
10) R

Chapter 11:

4.

A to B)

Arithmetic mean:
System A: (45 + 300 + 240 + 400 + 800) / 5 = 357
System B: (125 + 275 + 100 + 300 + 1200) / 5 = 400
Geometric mean:
System A: (45 x 300 x 240 x 400 x 800)^(1/5) = 253.01076
System B: (125 x 275 x 100 x 300 x 1200)^(1/5) = 262.12536

B to C)

Arithmetic mean:
System B: (125 + 275 + 100 + 300 + 1200) / 5 = 400
System C: (75 + 350 + 200 + 500 + 700) / 5 = 365
Geometric mean:
System B: (125 x 275 x 100 x 300 x 1200)^(1/5) = 262.12536
System C: (75 x 350 x 200 x 500 x 700)^(1/5) = 283.69094

A to C)

Arithmetic mean:
System A: (45 + 300 + 240 + 400 + 800) / 5 = 357
System C: (75 + 350 + 200 + 500 + 700) / 5 = 365
Geometric mean:
System A: (45 x 300 x 240 x 400 x 800)^(1/5) = 253.01076
System C: (75 x 350 x 200 x 500 x 700)^(1/5) = 283.69094

It seems very surprising that system B is shown to take more time than system C using the arithmetic mean, but it is shown that system C takes more time than system B using the geometric mean. System C is significantly faster than system B running program z (500 seconds faster). The difference between the arithmetic comparison and the geometric comparison is likely due to small values having an inordinate amount of affect on the geometric mean comparison's outcome (program w, x, and y run somewhat slower on system C than B).

It also seems surprising that system C is shown to be somewhat significantly slower than system A by the Geometric comparison. System C is actually faster than system A while running several programs. The difference is likely the result of the same reason mentioned above.

8. <u>Model Q has disk drives with an average access time of 12ms, while Model S has 15ms disk drives. Therefore Model Q will perform 25% better than Model S.</u>

I would use the data I have from Model S with Amdhal's Law to find the overall speedup for Model Q in relation to Model S because a speedup in the disk access time will only change the entire system speed based on the amount of disk access in relation to the entire system.

The Speedup would be: 1 / [(1 - %DiskUsageModelS) + (%DiskUsageModelS/1.25)]

I would use this speedup as the recorded performance metric for Model Q.

16. Probe reports: PC, IR, AC, MAR, MBR. 1GHz clock. Each cycle, status of each register (5 total) is written to memory. Each register is 64 bits wide. Storage required for 2 seconds of probing?

1 GHz is 1,000,000,000 cycles per second, so 2 seconds will equal 2,000,000,000 clock cycles. Each register is 64 bits wide, and there are 5 registers so each cycle will store 320 bits (the state of each register). So [320 * 2,000,000,000] = 640,000,000,000 bits are required for storage. There are 8589934592 bits in a byte, so the storage required in gigabytes is approximately 74.5058 gigabytes.

18. Read write head begins at track 50, moving towards the outer tracks.

Tracks for Algorithms

Head begins at 50 moving to lower. For each.								
FCFS (order)	54	36	21	74	46	35	26	67
Tracks:	4	18	15	53	28	11	9	41
							Total	**179**
Requests	54	36	21	74	46	35	26	67
SSTF (order)	54	46	36	35	26	21	67	74
Tracks:	4	8	10	1	9	5	46	7
							Total	**90**
Requests	54	36	21	74	46	35	26	67
SCAN (order)	54	67	74	46	36	35	26	21
Tracks:	4	13	7	28	10	1	9	5
							Total	**77**
Requests	54	36	21	74	46	35	26	67
LOOK (order)	46	36	35	26	21	54	67	74
Tracks:	4	10	1	9	5	33	13	7
							Total	**82**

```
25. Microprocessor requires 2, 4, 8, 12, 16 machine cycles to perform operations.
17.5% require 2 cycles, 12.5% require 4 cycles, 35% require 8 cycles, 20% require 12
cycles, 15% require 16 cycles.

a) Average: (.175*2 + .125*4 + .35*8 + .2*12 + .15*16) = 8.45 cycles/second
b) 1 MIPS means 1 million instructions per second, so it would have to have an 8.45MHz
   clock rate to be a 1 MIPS machine (8.45MHz is 8.45 million cycles per second, the
   average instruction cycle count).
c) [30% of normal cycle execution with 16 extra cycles] * [70% normal execution]
             .30*(.175*(2+16)+.125*(4+16)+.35*(8+16)+.2*(12*16) + .15*(16+16))
                                              +
                        .70*(.175*2+.125*4+.35*8+.2*12+.15*16)
                           = 23.09 instructions per second.
```