UC Berkeley
College of Engineering
EECS Department

EECS150 J. Wawrzynek
Spring 2002

Quiz #10

Name: _____ Solution_____

SID: _____Lab section number: _____

Consider the design of a list processor similar to the one presented in class and in the homework. This circuit iteratively forms the sum of all the 8-bit integers stored in a linked-list structure starting at memory address 0. The integers are stored in 2 byte nodes (the first byte is a pointer to the next node and the second is the integer). Nodes are not restricted to be aligned on even multiples of 2 bytes.

Unlike the processor presented in class, this processor, in addition to forming the sum, stores to memory into each node the sum of all the nodes up to that point. Therefore, after completion, node *i* will contain the sum of the values of all the nodes from the first up to and including the *i*th. For example, the original list 1→ 2 → 3 → 4 will be transformed into 1→ 3 → 6 → 10.

Your circuit includes a memory with an 8-bit data input/output port, a single adder, multiplexors, and registers of your choice.

a) Draw a resource utilization chart for this list processor. Fill in the execution schedule showing at least three loop iterations. Attempt to create a schedule in such a way as to minimize 1) the total number of cycles needed to process a list, and 2) the number of cycles in the control loop.

   Use subscripts to distinguish different loop iterations.

One of the approaches is to enumerate the operations for a single node and then consider pipelining three iterations for the minimum number of cycles.

Operations for a single node:

1) *Fetch the address (Next) for the head pointer from memory (this step can be skipped if the head pointer is initialized at the beginning).*
2) *Calculate the address for the number location within the same node (NumA ← Next + 1).*
3) *Fetch the number from memory.*
4) *Add the value of the number to the sum and form a new sum.*
5) *Store the new sum back to the memory location of NumA*

Now we have a recourses utilization chart for operations for a single node as shown below:

Steps (cycles)	1	2	3	4	5
Memory (R/W)	Next		X		Store
Adder		NumA		Sum	

Next we need to include two more iterations for a minimum number of cycles. The idea is to pipeline the operations. So we could interleave operations between iterations and put a memory read/write and an adder operation in the same cycle as possible. The final resources utilization chart is shown below:

Cycles	1	2	3	4	5	6	7	8	9	10	11
Mem	Next1		X1	Next2	Store1	X2	Next3	Store2	X3		Store3
Adder		NumA1		Sum1	NumA2		Sum2	NumA3		Sum3	

The total number of cycles is 11, or 10 if the head pointer is initialized at the beginning. Notice that the number of cycles in the control loops is 3 (i.e, cycles 4, 5, and 6 as one loop).

Based on your resource initialization chart, write the RTL description of your solution.

Cycle 1: *Next ← Mem[Next]*
Sum ← Sum + X

Cycle 2: *Mem[NumA] ← Sum*
NumA ← Next + 1

Cycle 3: *X ← Mem[NumA]*