Brown University
School of Engineering
ENGN 164 Design of Computing Systems
Professor Sherief Reda
Homework 07. 140 points. Due Date: Monday May 12th in B&H 349

---

1. [30 points] Consider the non-pipelined implementation of a simple processor that executes only ALU instructions in the figure. The simple microprocessor has to perform several tasks. First, it computes the address of the next instruction to fetch by incrementing the PIC. Second, it uses the PC to access the I-cache. Then the instruction is decoded. The instruction decoder itself is divided into smaller tasks. First, it has to decode the instruction type, Once the opcode is decoded, it has to decode what functional units are needed for executing the instruction. Concurrently, it also decodes what source registers or immediate operands are used by the instruction and which destination register is written to. Once the decode process is complete, the register file is accessed (the immediate data are accessed from the instruction itself) to get the source data. Then the appropriate ALU function is activated to compute the results, which are then written back to the destination register. Note that the delay of every block is shown in the figure. For instance, it takes 6 ns to access I-cache , 4 ns to access register file, etc.

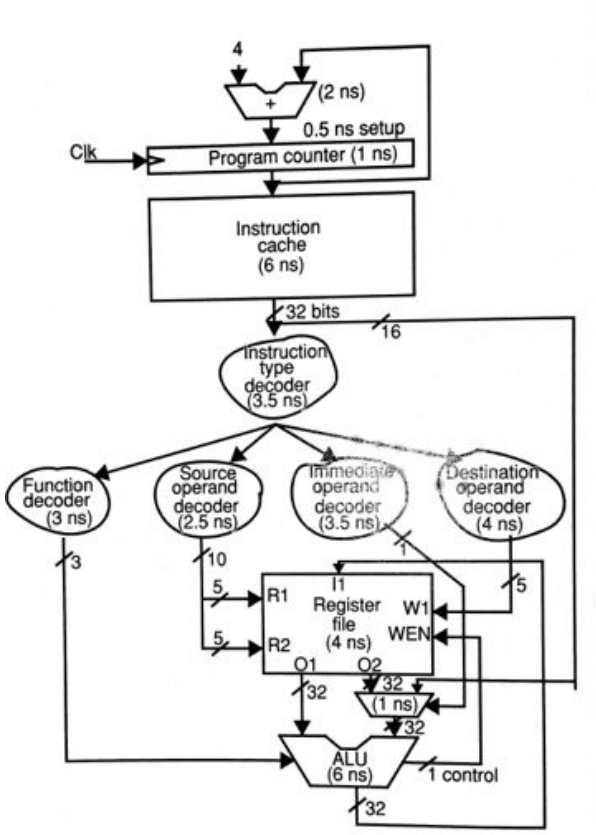
a. Generate a 5-stage (IF, ID1, ID2, EX, WB) pipelined implementation of the processor that balances each pipeline stage, ignoring all data hazards. Each sub-block in the diagram is a primitive unit that cannot be further partitioned into smaller ones. The original functionality must be maintained in the pipelined implementation. In other words, there should be no difference to a programmer writing code whether this machine is pipelined or otherwise. Show the diagram of your pipelined implementation.1

b. What are the latencies in (in nanoseconds) of the instruction cycle of the non-pipelined and pipelined implementations? Assume each pipeline register has a delay of 0.5 ns.

c. What are the machine cycle times (in nanoseconds) of the non-pipelined and the pipelined implementations?

d. What is the potential speedup of the pipelined implementation over the original non-pipelined implementation?

e. Which microarchitectural techniques could be used to reduced further the machine cycle time of pipelined design? Identify bottlenecks. Explain how the machine cycle time is reduced.



2. [25 points] The following program is executed on the classical 5-stage pipeline design. The program searches an area of memory and counts the number of times a memory work is equal to a key word:

SEARCH:	LW R5, 0(R3)	/I1 Load item
	SUB R6, R5, R2	/I2 Compare with Key
	BNEZ R6, NOMATCH	/I3 Check for match
	ADDI R1, R1, #1	/I4 Count for matches
NOMATCH:	ADDI R3, R3, #4	/I5 Next item
	BNE R4, R3, SEARCH	/I6 Continue until all items

Branches are predicted untaken always and are taken in EX if needed. Hardware support for branches is included in all cases. Consider several possible pipeline interlock designs for data hazards and answer the following questions for each loop iteration, except for the last iteration.

a. Assume first that the pipeline has no forwarding unit and no hazard detection units. Values are not forwarded inside the register file. Re-write the code by inserting NOPs wherever needed so that the code will execute correctly.

b. Next, assume no forwarding at all, but a hazard detection unit the stalls instructions in ID to avoid hazards. How many clock cycles does it take to execute one iteration of the loop (1) on a match and (2) on a no match?

c. Next, assume internal register forwarding and a hazard detection unit that stalls instructions in ID to avoid hazards. How many cycles does it take to execute one iteration of the loop (1) on a match and (2) on a no match?

d. Next, assume full forwarding and a hazard detection unit that stalls instructions in ID to avoid hazards. How many clocks does it take to execute one iteration of the loop (1) on a match and (2) on a no match?

e. A basic block is a sequence of instructions with one entry point and one exit point. Identify basic blocks (using instruction numbers) in the code. Is it safe for the compiler to move I5 up across the BNEZ. Does this help? How? Is it always safe to move instructions across basic block boundaries?

3. [25 points] We consider the execution of a loop in a single-cycle VLIW processor. Assume that any combination of instruction types can be executed in the same cycle. Note that such assumption only removes resource constraints, but data and control dependencies must be still handled correctly. Assume that register reads occur before register writes in the same cycle.

```
loop:  lw    $1, 40($6)
       add   $5, $5, $1
       sw    $1, 20($5)
       addi  $6, $6, 4
       addi  $5, $5, -4
       beq   $5, $0, loop
```

a. [20 points] Unroll this loop once and schedule it for a 2-issue static VLIW processor. Assume that the loop always executes an even number of iterations. Hint: for correct scheduling, you need to identify all potential data and control hazards. Use register renaming whenever possible to eliminate false dependencies.
b. [5 points] What is the speed-up of using your scheduled code instead of the original code? Assume that the loop will be executed a very large number of times.

4. [20 points] For a direct-mapped cache design with 32-bit address, the following bits of the address are used to access the cache. Tag: 31-12, Index: 11-6, and Offset: 5-0.

a. What is the cache line size (in words)?

b. How many entries does the cache have?

c. What is the ratio between the total bits required for such a cache implementation over the data storage bits?

Starting from the power on, the following byte-addressed cache references are recorded Address: 0, 4, 16, 132, 232, 160, 1024, 30, 140, 3100, 180, 2180.

d. How many blocks are replaced?

e. What is the hit ratio?

f. List the final state of the cache, with each valid entry represented as a record of <index, tag, data>.

5. [20 points] Consider a virtual memory system that can address a total of 2^{32} bytes but limited to 8 MB of physical memory. Assume that virtual and physical pages are each 4 KB in size.

a. How many bits is the physical address?

b. What is the maximum number of virtual pages in the system?

c. How many physical pages are in the system?

d. How many bits are the virtual and physical page numbers?

e. Suppose that you come up with a direct mapped scheme that maps virtual pages to physical pages. The mapping uses the least significant bits of the virtual page number to determine the physical page number. How many virtual pages are mapped to each physical page? Why is this "direct mapping" a bad plan?

f. Clearly, a more flexible and dynamic scheme for translating virtual addresses into physical addresses is required than the one described in part (d). Suppose you use a page table to store mappings (i.e., translations from virtual page number to physical page number). How many page table entries will the page table contain?

g. Assume that, in addition to the physical page number, each page table entry also contains some status information in the form of a valid bit (V) and a dirty bit (D). How many bytes long is each page table entry? (Round to an integer number of bytes.)

h. Sketch the layout of the page table. What is the total size of the page table in bytes?

6. [20 points] You decide to speed up the virtual memory system of the previous exercise by using a translation look aside buffer (TLB). Suppose your memory system has the characteristics shown in the given table. The TLB and cache miss rates indicate how often the requested entry is not found. The main memory miss rate indicates how often page faults occur.

a. What is the average memory access time of the virtual memory system before and after adding the TLB? Assume that the page table is always resident in the physical memory and is never held in the data cache.

b. If the TLB has 64 entries, how big (in bits) is the TLB? Give numbers for data (physical page number), tag (virtual page number), and valid bits of each entry. Show your work clearly.

Memory unit	Access Time (Cycles)	Miss rate
TLB	1	0.05%
cache	1	2%
main memory	100	0.0003%
disk	1,000,000	0%

c. Sketch the TLB. Clearly label all fields and dimensions.

d. What size SRAM would you need to build the TLB described in part (c)? Given your answer in terms of depth $\times$ width.