Assignment 3

Computer Architecture (2024-2025) December 15, 2024

George Theodoridis (S5951178)

INSTRUCTIONS

1. Submission is only via Brightspace. Deadlines are strict.
2. The exercises in this assignment add up to 100 points. To calculate your grade simply divide the number of points by 10.
3. You must submit a pdf typeset in (La)TeX (no handwritten solutions) using **this** template.
4. Seeking solutions from the internet, from any external resource, or from any other person is prohibited.
5. Please note that the course lecturer reserves the right to ask the student submitting the assignment to explain the answers to any or all questions. If the student is unable to provide a satisfactory answer then that question may receive partial/no credit.
6. Of course, university policies on plagiarism always apply. In particular, any suspected plagiarism will be reported to the Board of Examiners.

---

1. You are working on building your CPU. After working hard, you managed to implement 60 separate instructions (each with a separate OPCODE), and have decided that 32 registers should be plenty for all of them. Your computer will be taking 32 bits for each instruction, and uses the following table structure: **(10 points)**

OPCODE	SR	DR	IMM

The question is, what is the range of values that IMM is able to take (knowing that you decided to use 2's complement for it!), and how many OPCODEs are currently unused?

Solution: With 6 bits for the OPCODE, there are $2^6 = 64$ possible opcodes. Currently, 60 opcodes are being used.
Unused opcodes = $64 - 60 = 4$.

2. Assume you are building another computer! This time you have decided that the memory addressability is 64 bits. What does this tell you about the size of the MAR and MDR? **(10 points)**

Solution: the MAR is 64 bits because it must be as wide as the bus (holds the memory address), and the MDR is 64 bits (holds the data word and has the same width as the data-bus).

3. `LC-3` Instructions questions **(20 points)**

   a) How might one use a single `LC-3` instruction to move the value in `R2` into `R3`?

   b) The `LC-3` has no subtract instruction. How could one perform the operation `R1 = R2 - R3` using only three `LC-3` instructions?

   c) Using only one `LC-3` instruction and without changing the contents of the register, how might one set the condition codes based on the value that resides in `R1`?

   d) Is there a sequence of `LC-3` instructions that will cause the condition codes at the end of the sequence to be both `N=1`, `Z=1`, and `P=0`? Explain.

   e) Write the `LC-3` instruction that clears the contents of `R4`.

Solution:

a) ADD R3, R2, #0

b) NOT R3, R3
ADD R3, R3, #1
ADD R1, R2, R3

c) ADD R1, R1, #0

d) at the completion of any instruction that writes to a register, exactly one of the three condition bits (N, Z, or P) will be set, so no.

e) AND R4, R4, #0

4. Draw a finite-state machine for tennis scoring. The rules of tennis are as follows:

   To win, you need at least four points and you must have at least two points more than your opponent. Start with a state $(0, 0)$, indicating that no one has scored yet. Then add a state $(1, 0)$, meaning that `A` has scored. Label the arc from $(0, 0)$ to $(1, 0)$ with an `A`. Now add a state $(0, 1)$, indicating that `B` has scored, and label the arc from $(0, 0)$ with a `B`. Continue adding states and arcs until all the possible states have been included. **(20 points)**

Solution:

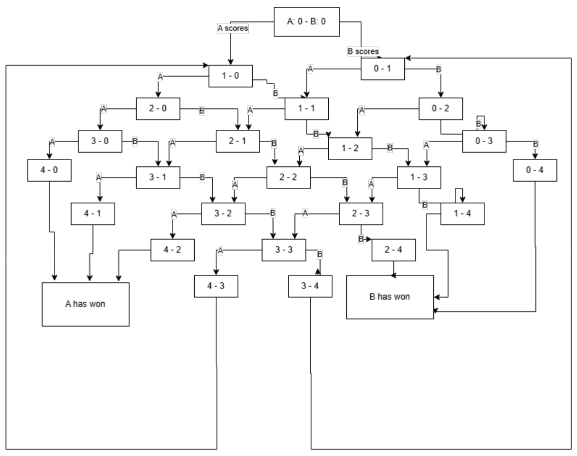

Figure 1: ex4

5. **Bus performance analysis exercise**
   In a computer system, three types of buses are commonly used:

   - **Data Bus**: Transfers actual data between components (e.g., CPU and memory).
   - **Address Bus**: Carries the address of the memory location or I/O device being accessed.
   - **Control Bus**: Carries control signals (e.g., read/write operations).

   Consider a system with the following specifications:

   - **Data Bus Width**: 16 bits.
   - **Address Bus Width**: 16 bits.
   - **Control Bus Width**: 2 bits.
   - Each bus has the same transfer rate, and each data transfer takes 10 clock cycles.
   - **Clock Speed**: 50 MHz.

   1. **Calculate the data transfer rate (in bytes per second) of the data bus.**
   2. **Determine how long it will take to read 10 MB (megabytes) of data from memory using the data, address, and control buses, assuming:**
      - The address and control buses transfer addresses and control signals only once per data transfer.
      - Ignore memory access delays.

   (15 points)

Solution:

- data-bus transfers per second: $50MHz/(10cycles/transfer) = 5 * 10^6$ Bytes transferred per second $= 5 * 10^6 transfers/s * 2bytes/transfer = 10 * 10^6 bytes$ so 10 MB/s
- each cycle = 20 ns, so 30 cycles = 30*20 ns = 600 ns 30 * 20 ns = 600 ns per 2 bytes.
- $wehave(10 * 10^6)/2 = 5 * 10^6 transferssototaltime = 5 * 10^6 transfers * 600ns/transfer = 3 * 10^9 ns = 3s.$

6. **Cache Performance Comparison**

   - **Cache size**: 64 KB
   - **Cache block size**: 32 bytes
   - **Cache access time**: 2 nanoseconds
   - **Memory access time**: 50 nanoseconds
   - The system cache is **direct-mapped**, and the CPU generates **32-bit memory addresses**.
   - The cache has a **hit ratio of 85%**.

   The CPU reads 10 MB of data from memory in a random access pattern, first checking the cache. If the data is not found in the cache, it is retrieved from memory.

   1. Calculate the total time required to read the data with the given hit ratio.
   2. Determine the time required to read the same data without using the cache.
   3. Compare the performance with and without the cache. Express the improvement as a percentage.

   (15 points)

Solution:

- average access time per byte $= (0, 85 * 2ns) + (0, 15 * 50ns) = 1.7ns + 7.5ns = 9.2ns. Total time = 10^7 bytes * 9, 2ns/byte = 9.2 * 10^7 ns = 0.092s.$
- every access goes to memory at 50 ns per byte, so total time is: $10^7 * 50 = 5 * 10^8 = 0.5s.$
- speedup = 0.5/0.092 = 5.43 (approximately). imporovement = (no-cache time - cache time)/ (no-cache time) * 100% = (0.5-0.092)/0.5 * 100% = 81.6 (approximately)

7. Consider a processor with a 4-stage pipeline (Fetch, Decode, Execute, Write-back). Each stage takes 10 nanoseconds to complete. Assume the following without pipelining, a single instruction takes 40 nanoseconds to complete (10 nanoseconds for each stage) **(10 points)**
   - Calculate the total time to execute 100 instructions with and without pipelining.
   - Determine the speedup achieved using pipelining.

Solution:

- without pipelining: Each instruction $= 4stages*10ns/stage = 40ns/instruction. For 100 instructions = 100 * 40ns = 4000ns = 4 microseconds$
  with pipelining: (pipeline depth + N - 1) * (stage time) = 30 ns + (100 - 1) * 10 ns = 40 ns + 990 ns = 1030 ns.
- speedup: time without pipelining/ time with pipelining = 4000/1030 = 3.88 (approximately)