**Week 4 Practical**

**Processors and Pipelining**

**Q1:** Postal Service.

* **a: What are the pipeline stages?**
  + Pick up from drop box.
  + Deliver to sorting warehouse.
  + Sort by fields (Address, Name, Postcode etc.)
  + Post via delivery pipeline (Car/Van/Bike for local, Truck for non-local, plane or ship for international.
  + Arrival at shipping address.
* **a: How long does each task take?**
  + Within a few hours or the next day.
  + A few hours.
  + One or two days.
  + From one day to a few weeks or months.
* **c: Are there any situations that could cause a “stall”?**

Yes, many. Any of the vehicles or the machinery used to sort mail could break down, a driver could have a personal issue (health, dependents etc.). None of these would happen frequently though and are not likely to stall the process often.

**Q2:** Preparing food in advance for service in a restaurant e.g., cooking burgers/fries to be ready for orders, preparing uncooked food ready to be cooked quickly.

* **a: How much time does it save?**
  + A lot. A burger could take 10 to 15 minutes to cook, but if there are some ready on the grill, they make take less than five minutes to prepare. Preparing other uncooked, or cold foods that are ready cook takes a huge amount of time out of the process for example: Dicing onions, preparing pasta, cooking tomato sauce etc.
* **b: Can the speculative activity fail?**
  + Yes. The ingredients may not be prepared correctly or ruined while being cooked. They may also not taste as they should. Another issue is preparing too much food and need to throw it out, wasting money.

**Q3:**

**1:** Explain each basic CPU configuration:

* **ALU:** Arithmetic Logic Unit. Perform all arithmetic and logic functions (Integer, Float etc.)
* **MDR:** Memory Data Register. A memory cache that stores and reads data from memory.
* **CU:** Control UNIT. Recognizes the current instruction and executes it to the appropriate device.
* **IR:** Instruction Register. Decodes and stores the next instruction the be executed by the CU.
* **MAR:** Memory Address Register. Stores the address of the next piece of data to be accessed from memory.
* **ACC:** Accumulator. A register that stores the results from the ALU.
* **PC:** Program Counter. Stores the address of the next instruction to be accessed from memory by the MAR.

**2:** Describe how the fetch-decode-execute-store cycle works:

* Fetch: Address is given to MAR by PC. MAR indicates which data to be accessed by the MDR. MDR accesses data and passes it to the IR.
* Decode: Instruction is decoded by the IR ready to be utilized by the CU.
* Execute: The CU analyzes the instruction and sends it to the appropriate device to be executed.
* Store: The executed data is then stored on the MDR to be written to memory, or used again by the CPU.

**Q4:** CPU has a 4-stage pipeline, and each stage takes one clock cycle.

* a: 1.
* b:
* c: 3.
* d: 5.
* e: n cycles.
* f:

A F D E stall

B F D stall stall

C F stall stall stall

D F D E S

**Q5:** Compare the benefits and drawbacks of:

* A deeply pipe-lined processor:
  + Can process many more instructions in parallel but can suffer from latency and “bubbles”.
* A multi-core processor:
  + Very high performance but cost a lot in materials, use more power, and generate more heat.
* Simultaneous multithreading:
  + Additional performance as pipelines can process two instructions by taking advantage of stall times.

**Q6:**

**1:** What is the difference between and CPU and a GPU?

* A CPU has between 1 and 16 cores and is designed to perform diverse and complex functions that can vary a lot and require flexible logic from the processor. A GPU has possibly hundreds of cores that are designed to process one, or a few, tasks over and over so cores can be specialized to process a few tasks with greater efficiency.

**2:** What is the difference between a GPU and a Graphics card?

* A graphics card is the complete assembly of a graphics unit. It involves the PCB, GPU, buses, cooling, and memory. The GPU refers to the processor of the Graphics Card.