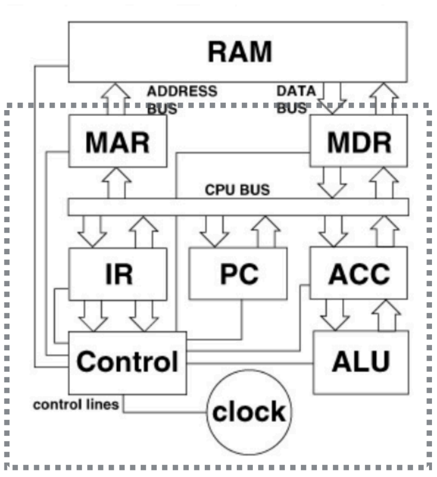
Solutions to Week 5 Processors and Pipelining

Question 1

The Central Processing Unit (CPU) is made up by the control unit, the arithmetic logic unit(ALU) and several registers.



1. Explain the functions of each basic CPU configuration — ALU, MDR, CU, IR, MAR, Accumulator, and Program Counter.
2. The main job of the CPU is to execute programs using the fetch-decode-execute cycle (also known as the instruction cycle). Describe how the fetch-decode-execute cycle works and show how addresses and data (instructions and operands) are transferred between memory and CPU.

Answer:

- **Program Counter** is used to store the address of the next instruction which is to be executed.
- **Instructions Register (IR)** is used to holds the actual encoded instruction being executed.
- **Memory Data Register (MDR)** holds the data being transferred to or from the memory location by the CPU.
- **Memory Address Register (MAR)** is used to store the address field of the data to be fetched to the CPU.
- **Control Unit (CU)** supervises the transfer of information among the registers and the ALU and has to tell which operation has to be performed.
- **Arithmetic Logic Unit (ALU)** performs all arithmetic and logical operations on the data.
- **Accumulator (ACC)** plays an important role being used to store the result of a calculation performed by the ALU.

Fetch — The first step the CPU carries out is to fetch some data and instructions (program) from main memory then store them in its own internal temporary memory areas.

i) Address of the next instruction is transferred from PC to MAR;
ii) The instruction is located in memory;
iii) The instruction is copied from memory to MDR.

Decode — The CPU is designed to understand a specific set of commands. These are called the 'instruction set' of the CPU. Each make of CPU has a different instruction set. The CPU decodes the instruction and prepares various areas within the chip in readiness of the next step.

i) The instruction is transferred to and decoded in the IR.

Execute — This is the part of the cycle when data processing actually takes place. The instruction is carried out upon the data (executed). The result of this processing is stored in yet another register.

i) Control unit sends signals to appropriate devices to cause execution of the instruction.

ii) Once the execute stage is complete, the CPU sets itself up to begin another cycle once more.

Question 2

Think of a real-world (non computing) activity that is analogous to instruction level parallelism; for example, a manufacturing assembly line, or a chain of events such as a supply-chain.

3. In this activity, what are the pipeline stages?
4. How long (comparatively speaking) does each stage take?
5. Are there any situations where a stage could "stall"?

Answer:

Consider a factory which produces bicycles. Each bicycle produced goes through five consecutive stages of production

1) adding the front wheel,
2) adding the handle-bar,
3) fitting the gear stick,
4) adding the rear wheel, and
5) fitting the mud guards.

We assume that each such bicycle part is available in stock. Assume that each of the five stages of production takes 6 minutes. Then it is simple arithmetic to conclude that — from the first stage to the last — it takes a total of 30 minutes to fully assemble a bicycle. But it should also be equally simple to conclude that the factory produces one bicycle every 6 minutes.

In fact, every stage finishes its share of the work on one bicycle in 6 minutes, passes that bicycle on to the next stage, and receives the next bicycle from the preceding stage. For 'clogfree' working of such a pipeline, it is necessary that each stage finished its share of the work in exactly 6 minutes — i.e. that the stage remain **synchronised.**

To ensure that no stage of production remains idle, we must ensure that the first stage of production begins work on a new bicycle once every 6 minutes. Suppose that — for some reason — the first stage remains idle for one time slot of 6 minutes. Clearly then, in the subsequent consecutive time slots, this idle time of 6 minutes — which we may call a 'bubble' — will pass through each of the remaining four stages of the pipeline.

Question 3

Assume a CPU has a 4-stage pipeline (i.e. stages for Fetch, Decode, Execute, Store) and each stage takes 1 clock cycle. Assume a stream of instructions A, B, C, etc...

Instruction No.	Pipeline Stage						
A	F_A	D_A	E_A	S_A			
B		F_B	D_B	E_B	S_B		
C			F_C	D_C	E_C	S_C	
D				F_D	D_D	E_D	S_D
E					F_E	D_E	E_E
F						F_F	D_F
Clock Cycle	1	2	3	4	5	6	7

1. How many instructions have been executed at the end of the 3rd clock cycle?

   Answer: 1, A
2. What is the state of the pipeline?

   Answer: F_C, D_B, and E_A
3. How many instructions have been executed at the end of the 5th clock cycle?

   Answer: 3 — A, B, and C
4. How many instructions have been executed at the end of the 7th clock cycle?

   Answer: 5 — A, B, C, D, and E
5. How many clock cycles does it take to fill an N-stage pipeline?

   Answer: N
6. Assume instruction A is a WHILE LOOP test, and instructions B & C are inside the loop and have been speculatively loaded into the pipeline. But on executing A, the processor

discovers the loop will terminate so B & C should not be performed. Draw the pipeline for the next 4 clock cycles.

Answer:

Instruction No.	Pipeline Stage							
A	F_A	D_A	E_A	**Branch!**				
B		F_B	D_B	**Flush**				
C			F_C	**Flush**				
D				F_D	D_D	E_D	S_D	
E					F_E	D_E	E_E	S_E
F						F_F	D_F	E_F
G							F_G	D_G
H								F_H
Clock Cycle	1	2	3	4	5	6	7	8

Question 4

Compare the relative benefits and drawbacks of:

1. A deeply pipelined processor
2. A multi-core processor
3. A processor core that supports simultaneous multithreading

Answer:

A deeply pipelined processor

Advantages

- Reduce clock cycles per instruction (CPI) and improve theoretical throughput

Disadvantages

- Problems with heat dissipation.
- Other parts of the processor e.g. memory might not be able to keep up.
- Potentially inefficient pipeline from stalls, etc.
- A deeply pipelined processor with poor branch prediction will potentially have less performance that a shallow pipelined processor at a lower clock speed

A multi-core processor (for example duo-core and quad-core)

Advantages

- Better at multitasking or parallelisable workloads

Disadvantages

- Problems in expense (more silicon needed to build chip)
- Internal complexity to ensure coherency between cores.
- **Not useful** for single non-parallel workloads

A processor core that supports simultaneous multithreading (SMT)

A **thread** is a basic unit of CPU utilisation. You can have a traditional (heavyweight) single-threaded process (since it is similar to a real process but executes within the context of a process and shares the same resources allotted to the process by the kernel) or a multi-threaded process.

Simultaneous multithreading (SMT): Issue multiple instructions from multiple threads in one cycle. The processor must be superscalar to do so.

Advantages

- Utilises core more efficiently — Responsiveness (allow one thread of program to run whilst another thread is blocked) and Resource sharing (threads share the same data/code as the process).
- Economy — reduces context switching (note: context switching of processes compared to context switching of threads)
- Generally faster throughput in terms of total workload.
- Scalability – benefits of multithreading can be greatly increased in a multiprocessor architecture, where threads maybe running in parallel on different processors.

Disadvantages

- Depending on the design and architecture of the processor, SMT can decrease performance if any of the shared resources are bottlenecks for performance.
- Critics argue that it is a considerable burden to put on software developers that they have to test whether simultaneous multithreading is good or bad for their application in various situations and insert extra logic to turn it off if it decreases performance.
- There is also a security concern with certain simultaneous multithreading implementations. Intel's hyper-threading implementation has a vulnerability through which it is possible for one application to steal a cryptographic key from another application running in the same processor by monitoring its cache use.

Question 5

Central Processing Unit Architecture operates the capacity to work from "Instruction Set Architecture" to where it was designed. The architectural designs of CPU are RISC (Reduced instruction set computing) and CISC (Complex instruction set computing).
There was a time when RISC architecture threatened to overtake the more traditional CISC architecture. Investigate the properties of RISC and CISC and why this did not happen in the end.

Answer:

CISC Advantages:

- Compiler has to do very little work to translate a high-level language statement into assembly.
- Length of the code is relatively short
- Very little RAM is required to store instructions.
- The emphasis is put on building complex instructions directly into the hardware.

RISC Advantages:

- Each instruction requires only one clock cycle to execute, the entire program will execute in approximately the same amount of time as the multi-cycle "MULT" command.
- These RISC "reduced instructions" require less transistors of hardware space than the complex instructions, leaving more room for general purpose registers. Because all of the instructions execute in a uniform amount of time (i.e. one clock)
- Pipelining is possible.

CISC	RICS
Emphasis on hardware	Emphasis on software
Includes multi-clock	Single-clock
Complex instructions	**Reduced instruction only**
Memory-to-memory: "LOAD" and "STORE" incorporated in instructions	Register to register: "LOAD" and "STORE" are independent instructions
high cycles per second, Small code sizes	Low cycles per second, **large code sizes**
Transistors used for storing complex instructions	Spends more transistors on memory registers
MULT A,B	LOAD R1,A LOAD R2,B PROD A,B S STORE R3, A

Question 6

What's the difference between a CPU and a GPU?

What are the differences between GPUs and graphics cards?

Answer:

CPUs are general purpose microprocessors while GPUs are application-oriented processors. That means, the instruction set used in CPUs is common to all x86-architecture processors regardless of whether they are made by Intel orAMD. GPUs on the other hand are basically CPUs with highly customised architectures that make them more suitable for visual processing than for general compute.

The CPU is the chip with many billions of transistors designed to do logic processing really quickly in order to perform all the actions the programs running on the computer require it to do, it may constitute a handful of "cores" as if there are multiple CPUs working alongside one another.

Technically GPU refers to a small chip on graphics card or integrated into the mainboard / CPU. This distinction is noticeable for dual-GPU graphics cards (graphics cards that have two GPUs on one card). A graphics card implies the term is about an expansion card.

The GPU is the chip which constitute a great many separate minimalist "cores". Each of these are much less capable than any of the cores in the CPU, but they're designed specifically to do certain tasks very well and they're designed to work together on the same task instead of each their own. These are then used to more quickly perform the calculations needed to display things like 3D graphics than the CPU would itself be able to do.