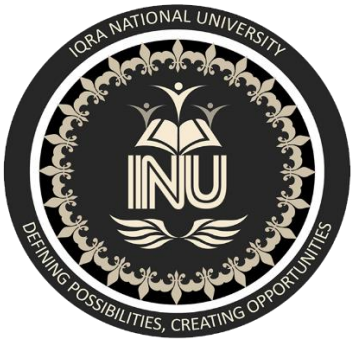
:: ASSIGNMENT # 2 ::

ID: 11533

Name: Ashir Ali Khan

Subject: Computer Architecture

Teacher: Sir Muhammad Amin



Iqra National University

Assignment : 2 :-

Question : 1 :-

Part : A :-

Answer :-

Different desktops applications that require the great power of contemporary microprocessor-based systems are :-

- Image processing applications.
- Speech recognition
- Video conferencing.
- Multimedia authoring.
- Voice and video annotation of files.
- Simulation modeling.

Part : B :-

Answer :-

The techniques used for contemporary processor to increase speed are :

1. **Pipelining :-**
   Pipelining is when computers receives multiple instructions and carry them out as they are received.
2. **Branch Prediction :-**
   Branch prediction is the process of being able to predict the next set of instructions so that they can carried out.
3. **Superscalar Execution :-**
   Superscalar execution is when you are able to give more than one set of instruction at time.

4. Data Flow Analysis :-
Data flow analysis analyzes instructions that need each other.

5. Speculative execution :-
Speculative execution carry out instructions before they are actually executed.

X——————X

Part : C :-

Answer :-
As clock speeds and logic density increase, a number of obstacles become more significant including...

- Power :-
The power density increases with an increase with an increase in logic density and clock speed. One Challenge of this is the difficulty of dissipating the heat generated on high-density, high-speed chips.
- RC delay :-
The speed at which electrons can flow on a chip between transistors is limited by the resistance and capacitance of the metal wire connecting them. Delay increases as the RC product increases. As components on the chip decrease in size, the wires are closer together, increasing capacitance.
- Memory latency :-
Memory speeds lag processor speeds.

X——————X

Part : D :-

Answer :-

Amdahl's law deals with the potential speedup of a program using multiple processors compared to a single processor. Consider a program running on a single processor such that a fraction $(1-f)$ of the execution time involves code that is inherently sequential, and a fraction f that involves code that is infinitely parallelizable with no scheduling overhead. Let T be the total execution time of the program using a single processor. Then the speed up using a parallel processor with N processors that fully exploits the parallel portion of the program is as follows.

$$\text{Speedup} = \frac{\text{Time to execute program on a single processor}}{\text{Time to execute program on N parallel processor}}$$

$$= \frac{T(1-f) + Tf}{T(1-f) + \frac{Tf}{N}} = \frac{1}{(1-f) + \frac{f}{N}}$$

Two important conclusions can be drawn:

1) When f is small, the use of parallel processors has little effect.
2) As N approaches infinity, speedup is bound by $1/(1-f)$, so that there are diminishing returns for using more processors.

Part : E :-

Answer :-

Multicore :-

Multicore refers to an architecture in which a single physical processor incorporates the core logic of more than one processor. A single integrated circuit is used to package or hold these processors. These single integrated circuit are known as a die. Multicore architecture places multiple processors cores and bundles them as a single physical processors. The objective is to create a system that can complete more tasks at the same time, thereby gaining better overall system performance. This technology is most commonly used in multicore processors, where two or more processors chips or cores run concurrently as a single system. Multicore-based processors are used in mobile devices, desktop, work stations and servers. The concept of multicore technology is mainly centered on the possibility of parallel computing, which can significantly boost computer speed and efficiently by including two or more central processing units (CPUs) in a single chip. This reduces the system's heat and power consumption. This means much better performance with less or the same amount of energy.

MIC :-

chip manufacturers are now in the process of making a huge leap forward in the number of cores per chip, with more than 50 cores per chip. The leap in performance as well as the challenges in developing software to exploit such a large number of cores has led to the introduction of a new term known as many integrated core (MIC).

GPGPU :-

A general-purpose GPU (GPGPU) is a graphics processing unit (GPU) that performs non-specialized calculations that would typically be conducted by the CPU (central processing unit). Ordinarily, the GPU is dedicated to graphics rendering.

GPGPUs are used for tasks that were formerly the domain of high-power CPUs, such as physics calculations, encryption, scientific computations and the generation of crypto currencies such as Bit coin. Because graphics cards are constructed for massive parallelism, they can dwarf the calculation rate of even the most powerful CPUs for many parallel processing tasks. The same shader cores that allows multiple pixels to be rendered simultaneously can similarly process multiple streams of data

at the same time. Although a shader core is not nearly as complex as a CPU, a high-end GPU may have thousands of shader cores; in contrast, a multicore CPU might have eight or twelve cores.

X——————X

Question :- 2 :-

Part : A :-

Given :-

Clock speed of the processor = 60 MHz

Number of instructions the executed program consist = 104,000

Instruction type	Instruction Count	Cycles per instruction
Integer arithmetic	46000	1
Data Transfer	330000	2
Floating point	160000	2
Control transfer	90000	2

To Find :-

CPI = ?

MIPS rate = ?

Execution time = ?

Solution :-

Calculating the CPI is :

$$CPI = \frac{\text{Instruction count} \times \text{Cycles per second}}{\text{Number of instruction the executed program consist.}} \quad \text{— (1)}$$

Substitute the values of "instruction count" and "cycles per second" from the above table in

equation (1).

$$CPI = \frac{(46000 \times 1) + (33000 \times 2) + (16000 \times 2) + (9000 \times 2)}{104,000}$$

$$CPI = \frac{46000 + 66000 + 32000 + 18000}{104,000}$$

$$= \frac{162,000}{104,000}$$

$$= 1.55.$$

Therefore, the CPI for this program is 1.55.

Now Calculating MIPS :-

$$MIPS = \frac{f}{CPI \times 10^6}.$$

Frequency is given as 60 MHz, putting values.

$$MIPS = \frac{60 \times 10^6}{1.55 \times 10^6}$$ → converted MHz to Hz.

$$= 38.70$$

Calculating execution time :-

Execution time = CPI x Instruction count x clock time

$$= \frac{CPI \times \text{Instruction count}}{\text{frequency}}.$$

$$= \frac{1.55 \times 104000}{60 \times 10^6}$$

$$= 0.0026$$

Execution time = 0.0026.

X——————X

Part : B :-

Answer :-

Given :-

Instruction type	Instruction Count (millions)	Cycles per instruction
Machine A.		
Arithmetic and logic	8	1
load and store	4	3
Branch	2	4
others.	4	3
Machine A		
Arithmetic and logic	10	1
load and store	8	2
Branch	2	4
others.	4	3

Solution :-

$$CPI_A = \frac{\sum CPI_i \times I_i}{I_c} = \frac{(8\times1 + 4\times3 + 2\times4 + 4\times3)\times10^6}{(8+4+2+4)\times10^6}$$

$$= 2.22.$$

$$MIPS_A = \frac{f}{CPI_A \times 10^6} = \frac{200\times10^6}{2.22\times10^6} = 90$$

$$CPU_A = \frac{I_c \times CPI_A}{f} = \frac{18\times10^6 \times 2.2}{200\times10^6} = 0.2s$$

$$CPI_B = \frac{\sum CPI_i \times I_i}{I_c} = \frac{(10\times1 + 8\times2 + 2\times4 + 4\times3)\times10^6}{(10+8+2+4)\times10^6}$$

$$= 1.92.$$

$$CPU_B = \frac{I_c \times CPI_B}{f} = \frac{24 \times 10^6 \times 1.92}{200 \times 10^6} = 0.23s.$$

X——————X

Part: C :-

Answer :-

(a) What is the relative size of the instruction count of machine code for this benchmark program running on the two machines?

The MIPS rate could be computed as the following:

$$[(\text{MIPS rate}) / 106] = \frac{I_c}{T}$$

$$I_c = T \times [(\text{MIPS rate}) / 10^6]$$

Now by computing the ratio of the instruction count of the IBM RS/6000 to the VAX 11/780 which is:

$$[X \times 18] / [12X \times 1] = 18X / 12X = 1.5.$$

b) What are the CPI values for the two machines?

Regarding to the VAX 11/780,

the CPI $= (5MH_z) / (1 MIPS) = 5.$

Regarding to the IBM RS/6000, the CPI

$= (25MH_z) / (18MIPS) = 1.4.$

X——————X

Part : D :-

Answer :-

(A) Determine the average CPI :-

Since we have the same instruction mix, that means the additional instructions for each task could be allocated appropriately between the instruction types. Therefore, the following table be gotten :

Instruction type	CPI	Instruction Mix
Arithmetic and logic	1	60%
load/store with cache hit	2	18%
Branch	4	12%
Memory reference with cache miss	12	10%

The average CPI = $(1 \times 0.6) + (2 \times 0.18) + (4 \times 0.12) + (12 \times 0.1) = 2.64$

Therefore, the CPI has been increased since the time for memory access is also increased

(B) Determine the average MIPS rate :-

MIPS = $400/2.64 = 152$

MIPS = 152

There is a corresponding drop in the MIPS rate.

(C) Calculate the speed up factor :-

The speedup factor equals to the ratio of the execution times. The execution time is calculated as the following:

$T = Ic / (MIPS \times 10^6)$.

For the one processor, $T_1 = (2 \times 10^6) / (178 \times 10^6)$

$= 11ms$.

For the 8 processor, each processor executes 1/8 of the 2 million instruction plus the 25,000

$$T_8 = \frac{\frac{2 \times 10^6}{8} + 0.025 \times 10^6}{152 \times 10^6} = 1.8ms$$

Therefore we have:

$$\text{Speedup} = \frac{\text{time to execute program on single processor}}{\text{time to execute program on N parallel processor}}$$

$$= \frac{11}{1.8} = 6.11.$$

(d) Compare the actual speedup factor with the theoretical speedup factor determined by Amdahl's law.

In fact, there are two inefficiencies in the parallel system.

The first one is that there are more additional instructions which is added to coordinate between threads.

The second one is that there is contention for memory access. Thus, none of the code

is inherently serial, and all of it is parallelizable but with scheduling overhead. It could be said that the memory access conflict means some extent memory reference instructions are not parallelizable.
By depending on the information given, it is not obvious how to quantify this effect in Amdahl's equation. Therefore, if it is supposed that the fraction of code, which is parallelizable, is $F = 1$, then Amdahl's law decreases to speed up $= N = 8$. Therefore, the actual speedup is only about 75% of the theoretical speed up.

X——————X