Assignment 1 - John Akujobi

Owner	John Akujobi
Type	Homework
Created time	@January 6, 2024 3:22 PM
Status	In progress

Question 1.4

Assume a color display using 8 bits for each of the primary colors (red, green, blue) per pixel and a frame size of 1280 × 1024.

a. What is the minimum size in bytes of the frame buffer to store a frame?

1. **Calculate the Total Number of Pixels**:
    - The resolution of the display is 1280 × 1024. This means there are 1280 pixels in each row and 1024 pixels in each column.
    - Total number of pixels = number of pixels in each row × number of pixels in each column.

    ```
    Total number of pixels = 1280 * 1024
    = 1310720
    ```

2. **Calculate the Bits Per Pixel (bpp)**:
    - The display uses 8 bits for each primary color (red, green, blue). Since there are 3 primary colors, the total bits per pixel is 8 bits/color × 3 colors.

    ```
    bits per pixel = 8 * 3
    = 24
    ```

3. **Calculate the Total Bits Required**:
    - Total bits required = total number of pixels × bits per pixel.

    ```
    Total bits required = 1310720 * 24
    31457280 bits
    ```

4. **Convert Bits to Bytes**:
    - Since 1 byte = 8 bits, we divide the total bits by 8 to get the total bytes required.

    ```
    31457280 bits/8 bits per byte
    = 3932160 bytes
    = 3840 kb
    = 3.75 mb
    ```

b. How long would it take, at a minimum, for the frame to be sent over a 100Mbit/s network?

```
Speed = 100,000,000 bits per second
Time = 31457280/100,000,000
= 0.3145728 sec
```

Question 1.5

Consider three different processors P1, P2, and P3 executing the same instruction set.

- P1 has a 3GHz clock rate and a CPI of 1.5.
- P2 has a 2.5GHz clock rate and a CPI of 1.0.
- P3 has a 4.0GHz clock rate and has a CPI of 2.2.

- Instruction Execution Rate IER = `Clock Rate / CPI (Cycles Per Instruction)`
- Number of Cycles NC= `Clock Rate * Execution Time`
- Number of Instructions NI = `Number of Cycles / CPI (Cycles Per Instruction)`

Processor	Clock Rate	CPI	IER = CR/ CPI	NC = CR * 10	NI = NC/CPI
P1	3 GHz	1.5	2.0 billion I/s	30 billion	20 billion
P2	2.5 GHz	1.0	2.5 billion I/s	25 billion	25 billion
P3	4.0 GHz	2.2	1.82 billion I/s	4 billion	18,181,818,182

a. Which processor has the highest performance expressed in instructions per second?

The highest performance comes from P2 with 2.5 billion instructions per second

b. If the processors each execute a program in 10 seconds, find the number of cycles and the number of instructions.

Processor	Clock Rate	CPI	NC = CR * 10	NI = NC/CPI
P1	3 GHz	1.5	30 billion	20 billion
P2	2.5 GHz	1.0	25 billion	25 billion
P3	4.0 GHz	2.2	4 billion	18,181,818,182

b. We are trying to reduce the execution time by 30%, but this leads to an increase of 20% in the CPI. What clock rate should we have to get this time reduction?

- Old Execution Time per Instruction ETI = `CPI / Clock Rate`
- New CPI = Original CPI × 1.2

- New Execution Time per Instruction = `Original Execution Time per Instruction × 0.7`
- New Clock Rate = `New CPI / New Execution Time per Instruction`

P	Clock Rate	CPI	ETI (Instructions per sec)	New CPI = CPI * 1.2	New ETI = ETI * 0.7	New CR = New CPI/New ETI
P1	3 GHz	1.5	0.5 e-9 seconds	1.8	0.35 e-9 seconds	5.143 GHz
P2	2.5 GHz	1.0	0.4 e-9 seconds	1.2	0.28 e-9 seconds	4.286 GHz
P3	4.0 GHz	2.2	0.55 e-9 seconds	2.64	0.385 e-9 seconds	6.857 GHz

Question 1.6

Consider two different implementations of the same instruction set architecture. The instructions can be divided into four classes according to their CPI (classes A, B, C, and D).

- P1 with a clock rate of 2.5GHz and CPIs of 1, 2, 3, and 3, and
- P2 with a clock rate of 3GHz and CPIs of 2, 2, 2, and 2

Given a program with a dynamic instruction count of 1.0E6 instructions divided into classes as follows:

- 10% class A
- 20% class B
- 50% class C, and
- 20% class D,

Which is faster: P1 or P2?

- Instruction Execution Rate IER = `Clock Rate / CPI (Cycles Per Instruction)`

a. What is the global CPI for each implementation

b. Find the clock cycles required in both cases.

Question 1.14

Assume a program requires the execution of

50 e6 FP instructions, 110 e6 INT instructions, 80 e6 L/S instructions, and 16 e6 branch instructions.

The CPI for each type of instruction is 1, 1, 4, and 2, respectively.
Assume that the processor has a 2GHz clock rate

Total execution time for the program:

Instruction Type	Number of Instructions	CPI	Total Cycles
FP	50,000,000	1	50,000,000
INT	110,000,000	1	110,000,000
L/S	80,000,000	4	320,000,000
Branch	16,000,000	2	32,000,000
Total			**512,000,000**

Execution Time (seconds): = total_cycles / clock_rate_Hz # in seconds

```
Execution Time (seconds)
= 512 e6/2 e9
= 0.256 seconds
```

1.14.1. By how much must we improve the CPI of FP instructions if we want the program to run two times faster?

```
New Target Execution time = 0.5 * Execution time
= 0.128 seconds
```

New Target Execution time = `256 e6/2 e9`

To make it run 2 times faster, This means we need to get the total clock cycles to be
Total Target Clock Cycles =
`256,000,000`

Reduce FP to 0

Now, lets say we totally reduce the FP Instruction type to 0 clock cycles

```
Total Clock cycles with FP at 0
= 512,000,000 - 50,000,000
= 462,000,000 clock cycles
```

This is still a lot higher than the target total of 256 e6 clock cycles needed to make the program 2 times faster.

So, modifying only the FP instructions will never get it to run twice as fast.

The maximum improvement that would give is 10% improvement

```
New total time with FP at 0
= 462 e6/2 e9
= 0.231 seconds

Difference = 0.256 - 0.231
= 0.025 sec

Percent = (0.025/0.256)
```

```
= 0.097 * 100
= 10% improvement
```

1.14.2. By how much must we improve the CPI of L/S instructions if we want the program to run two times faster?

We want to get the total cycles to 256 e6

L/S has 320 e6 clock cycles

- Total for other operations

  ```
  512 - 320 = 192 cycles
  ```

- Amount of L/S clock cycles that can be added

  ```
  Target - Other operations
  = 256 - 192
  = 64 e6 cycles

  The LS clock cycles have to be reduced to only 64 clock cycles
  ```

- Target CPI for L/S to get to 64 e6 cycles

  ```
  Instructions * new CPI = Clock cycles
  new CPI = Clock cycles/ Instructions
  = 64 e6/ 80 e 6
  = 0.8
  ```

We have to reduce the CPI for L/S operations by 80% down to 0.8 CPI in order to reduce the program time by half

1.14.3. By how much is the execution time of the program improved if the CPI of INT and FP instructions is reduced by 40% and the CPI of L/S and Branch is reduced by 30%?

Instructions	CPI	New CPI	Inst	Clock Cycles
INT	1	0.6	110000000	66000000
FP	1	0.6	50000000	30000000
L/S	4	2.8	80000000	224000000
Branch	2	2	16000000	32000000
				352000000

512 - 352 = 160 e6 clock cycles

160/512 = 0.3125

The execution time is improved by 31.25%

Or it improves it to 0.176 seconds

```
352 e6/2 e9
= 0.176s seconds
```

Question 1.15

When a program is adapted to run on multiple processors in a multiprocessor system, the execution time on each processor is comprised of computing time and the overhead time required for locked critical sections and/or to send data from one processor to another.

Assume a program requires t = 100 s of execution time on one processor. When run p processors, each processor requires t/p s, as well as an additional 4 s of overhead, irrespective of the number of processors. Compute the per-processor execution time for 2, 4, 8, 16, 32, 64, and 128 processors.

For each case, list the corresponding speedup relative to a single processor and the ratio between actual speedup versus ideal speedup (speedup if there was no overhead)

- Execution time per processor = `(t / p) + overhead`
- Speedup = `t / Execution time per processor`

Ratio

- Ideal speedup = `number of processors (p)`
    - = t / (t / p)
- Speedup Ratio = `Speedup / Ideal Speedup`

Number of Processors	Per-Processor Time (s)	Speedup (Relative to 1 Processor)	Ratio (Actual vs Ideal Speedup)
2	54.0	1.85	0.93
4	29.0	3.45	0.86
8	16.5	6.06	0.76
16	10.25	9.76	0.61
32	7.125	14.04	0.44
64	5.5625	17.98	0.28
128	4.78125	20.92	0.16

The speedup decreases as the number of processors increases. This is due to the overhead time, which remains constant per processor and becomes a significant part of the total execution time, especially with a higher number of processors. Thus, the actual speedup falls short of the ideal speedup.