Name: Lakhan Kumawat
Roll No: 1906055
Branch: CSE-1

Computer Architecture / CS4401 01/05/2021

Solution 1: Let a & b are hit ratios of level 1 and level 2 respectively

a)

Then the equation for access time can be written as follows.

$$T = T_1 + (1-a)T_2 + (1-a) \times (1-b)T_3$$

Also $T_1 = 50ns$

$T_2 = 200ns$ $\quad T \leq 100$

$T_3 = 5000ns$

on substituting a, b for the first case we get

$T = 50ns + (1-0.8)\,200ns + (1-0.8)(1-0.995) \times 5000\ ns$

(i) $T = 95ns$ for $a = 0.8$ and $b = 0.995$

$L_1 = 8M$ and $L_2 = 64M$

$T = 50 + (1-0.9) \times 200 + (1-0.9) \times (1-0.99) \times 5000\ ns$

(ii) $T = 75ns$ for $a = 0.9$ & $b = 0.99$

$L_1 = 16M$ and $L_2 = 4M$.

b) To find average value (i.e. access time) substitute value in equation)

1. $L_1 = 8M$, $a = 0.8$, $L_2 = 4M$, $b = 0.98$ So,

$T = 50 + 0.2 \times 200 + 0.2 \times 0.02 \times 5000$

$= 50 + 2 \times 20 + 4 \times 5$

$= 50 + 40 + 20$

$= 110\ ns.$

Name: Lakhan kumawat
Roll NO: 1906055
Branch: CSE-1
Course: CS4401
01/05/2021

2.

2. $L_1 = 16M$, $a = 0.9$, $L_2 = 16M$, $b = 0.09$ so,

$$T = 50 + 0.1 \times 200 + 0.1 \times 0.01 \times 5000$$

$$= 50 + 20 + 5 = 75\ ns$$

3. $L_1 = 64M$, $a = 0.95$, $L_2 = 64M$, $b = 0.995$

So, $$T = 50 + 0.05 \times 200 + 0.05 \times 0.005 \times 5000$$

$$= 50 + 5 \times 2 + \frac{5 \times 5 \times 5}{100}$$

$$= 50 + 10 + 1.25$$

$$= 61.25\ ns$$

$$T_{avg} = 50 + (1 - 0.90) \times 200 + (1 - 0.90)(1 - 0.98)\, 5000$$

$$= 50 + 20 + 10$$

$$= 80\ ns$$

Name: Lakhan Kumawat
Roll No: 1906055
Branch: CSE-1

Course: CS4401
01/05/2021

Solution 2. a)

$I_i \xrightarrow{WAW/WAR} I_j$

Here we follow hardware-based register rename policy.
We have following two approaches for handling.

1) If previous instructions ready to provide source operand values for I_i, as such instruction's execution completed then ~~RSFU~~ RS_{FU_i} to store all such values in its buffers so that to make I_j independent of I_j.

2) If previous instructions are not ready to provide source operands for I_i (as such instructions execution not completed) then RS_{FU_i} to store pointer to RS_s of a previous instructions to receive their values directly via CDB (Common Data Bus) to make I_i independent of I_j.

Solution 2. b)

Instruction Sequence

```
        BNEZ    R1      lbl
        ADD     R2      R3      R0
        ADD     R4      R5      R0
        J       lblL
lbl:    ADD     R6      R7      R0
        ADD     R8      R9      R0
lbl1: ...
```

For the given instruction sequence, we can used CMOVZ [DestReg.] [SrcReg] [CondReg.] as the predication instruction to generate the above instruction sequence.

Lakhan Kumawat
1906055
CSE-1

CS4401
01/05/2021

⇒

CMOVZ	R_2	R_3	R_1
CMOVZ	R_4	R_5	R_1
CMOVZ	R_6	R_7	R_1
CMOVZ	R_8	R_9	R_1

(3).

⇒

```
        BNEZ    R1   lb1
        ADD     R2   R3   R0
        ADD     R4   R5   R0
lbl:    J       lb11
        ADD     R6   R7   R0
        ADD     R8   R9   R0
lb11:
```

(2).

⇒

```
if (a=0)
{
    b=c;
    d=e;
}
else
{
   f=g;
   m=n;
}
```

(1).

Name: Lakhan kumawat
Roll No: 1906055
Branch: CSE

Course: CS4401
01/05/2021

Solution 3.) Mapping graphics pipeline to Graphics architecture :-

a)

A graphics pipeline or rendering pipeline is a conceptual model that describes what steps a graphics system needs to perform to render a 3D scene to a 2D screen.

Similarly Unified GPU architecture is based on parallel array of many programmable processors. They unify vertex, geometry and pixel shader processing and parallel computing on the same processors Unlike Circular GPU's which had separate processor dedicated to each processing type. The programmable processor array is tightly integrated with fixed function processors for texture filtering, rasterization and raster operations, anti-aliasing effects, compression and high definition video processing.

Although the fixed function processors significantly out perform more general programmable processor in terms of absolute performance constrained by an area, cost or power budget.

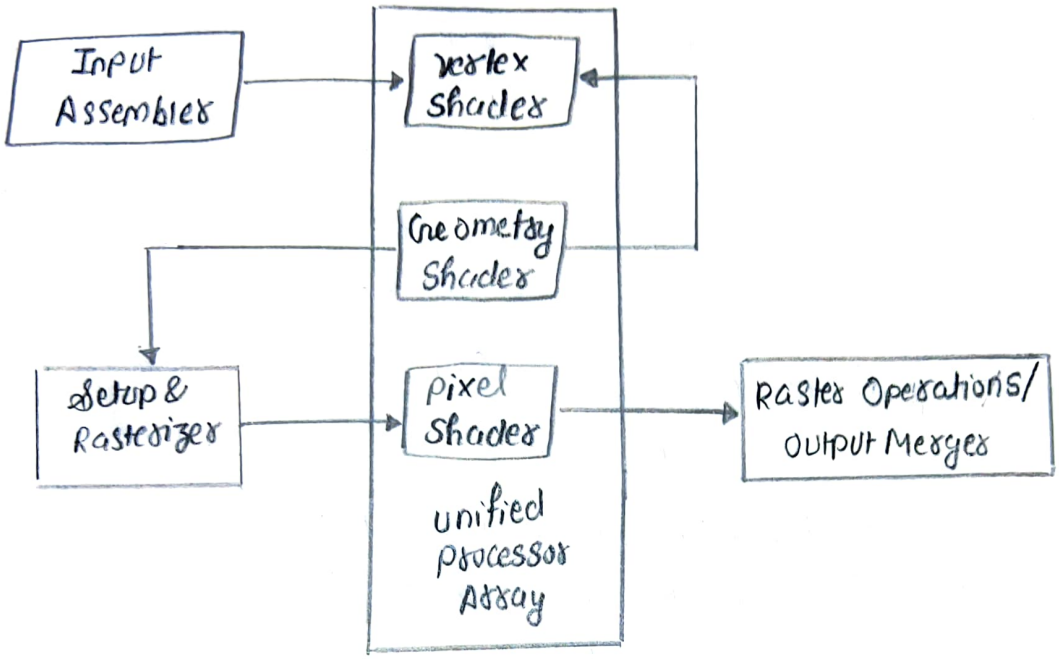
An unified GPU processor array contains many processors ones, typically organized into multithreaded multiprocessors

Name: Lakhan Kumawat
Roll NO: 1906055
Branch: CSE-1

6.

Course: CS4401
01/05/2021

Basic unified GPU architecture.

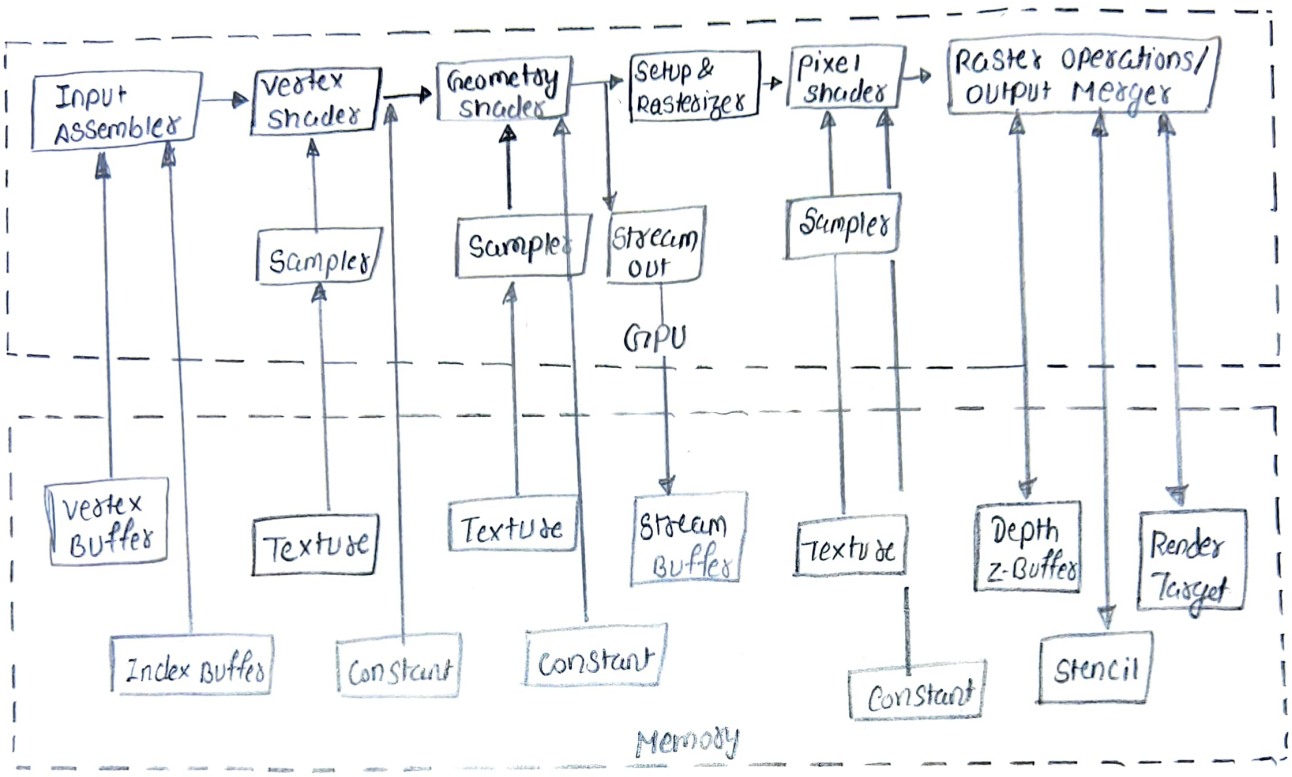


Name: Lokhan Kumawat
Roll No: 1906055
Branch: CSE-1

7.

Course: CS4401
01/05/2021

Solution 3) 10-Stage Graphics pipeline components in Direct3D interface with b) illustration.



Input Assembler Stage: The Direct3D 10 and higher API seperates functional areas of pipeline into stages; the first stage in the pipeline is input-assembler stage.

Vertex Shader stage: The vertex-shader stage processes vertices from the input assembler performing per-vertex operations such as transformations, skinning, morphing and per vertex lighting.

Geometry Shader: This stage runs application specified shader code with vertices as input and the ability to generate vertices on output.

~~Tessellation Stage~~

Stream output Stage: Purpose of this stage is to continuously output vertex data from the geometry-shader stage.

Name: Lakhan kumawat
Roll No: 1906055
Branch: CSE-1

Q. 8.

Course: CS4401
01/05/2021

Rasterizer Stage : This stage converts vector information into a rasterized image for the purpose of displaying real-time 3D graphics.

Pixel Shader Stage : It enables rich shading techiques such as per-pixel lightening and post-processing. A pixel shader is a program that combines constant variables, texture data, interpolated per-vertex values, and other data to produce per-pixel outputs. The rasterizer stage invokes a pixel shader once for each pixel.

Output-Merger Stage : It generates the final rendered pixel color using a combination of pipeline state, the pixel data generated by the pixel shader, the contents of the render targets and the contents of the depth/stencil buffers. The OM stage is the final step for determining which pixels are visible (with depth-stencil testing) and blending the final pixel colors.

Lakhan Kumawat
1906055
CSE-1
CS4401
01/05/2021

Solution 4).

a) We can find how many cores can be used for the 90% of the time when more than 54 are usable, as follows:

$$\text{Average processor usage} = 0.09 \times 50 + 0.01 \times 1 + 0.90 \times \text{Max processors}$$

$$54 = 0.9 \times 5 + 0.01 + 0.90 \times \text{Max processors}$$

$$54 = 4.51 + 0.90 \times \text{Max processor}$$

$$0.90 \times \text{Max processors} = 54 - 4.51$$

$$\therefore [\text{Max processor} = 55]$$

Solution) b) Speedup to be expected by A in comparison to any uniprocessor system.

$$\text{Speedup} = \frac{1}{\frac{\text{Fraction}_{55}}{55} + \frac{\text{Fraction}_{50}}{50} + (1 - \text{fraction}_{55} - \text{fraction}_{50})}$$

$[\because \text{Fraction}_{55} = 0.90, \ \text{Fraction}_{50} = 0.09]$

$$\text{Speedup} = \frac{1}{\frac{0.90}{55} + \frac{0.09}{50} + 0.0(1 - 0.90 - 0.09)}$$

$$= \frac{1}{\frac{9}{550} + \frac{9}{5000} + (1 - 0.99)}$$

$$= \frac{1}{0.016 + 0.0018 + 0.01}$$

$$= \frac{1}{0.0278}$$

Lakhan kumawat
1906055
CSE-1

01/05/2021

[Speedup = 35.9712]

Solution

c)

Now computing speedup on 24 processors

$$\text{Speedup} = \frac{1}{\frac{\text{Fraction}_{24}}{24} + (1 - \text{Fraction}_{24})}$$

$$= \frac{1}{\frac{0.99}{24} + 0.01}$$

$$= \frac{1}{0.0412 + 0.01}$$

$$= \frac{1}{0.0512}$$

$$= 19.53$$

Hence when considering both power constraints and Amdahl's law effects, the 96-processor version achieves less than a factor of 2 speedup over the 24-processor version.

In fact the speedup from clock rate increases nearly matches the speedup from the 4x processor count increase.