

Jaehoon Shim
Seongyeop Jeong
Ilkueon Kang
Wookje Han
Jinsol Park
(snucsl.ta@gmail.com)

Systems Software &
Architecture Lab.
Seoul National University

Fall 2022

4190.308:

Computer Architecture Lab. I



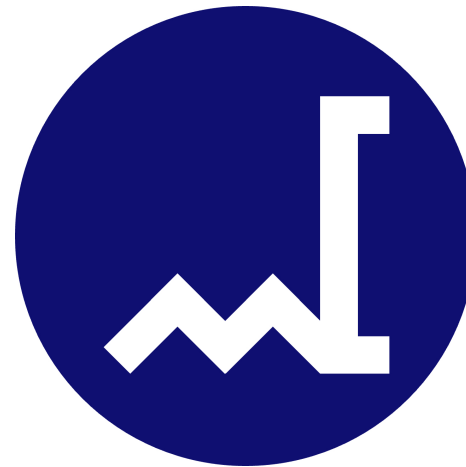
Image Compression

What is PNG Filtering?

- PNG file format supports a precompression step called filtering
- Filtering is a method of reversibly transforming the image data so that the main compression engine can operate more efficiently



JPEG, 188KB



PNG, 133KB

Simplified Image Compression

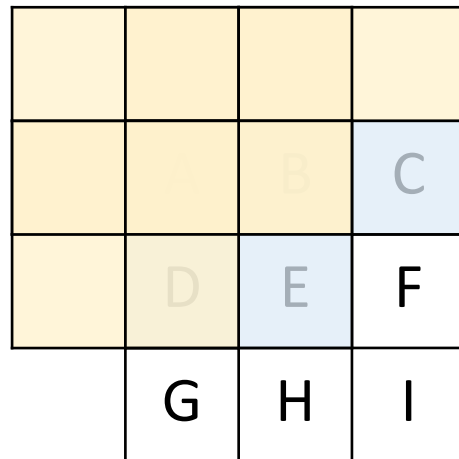
- The input will be a grayscale image
 - Pixel in grayscale image represents only an amount of light
 - Use unsigned integers with 8 bits per pixel (0~255)
- We will use a simplified Paeth filtering algorithm in the PNG format
 - Paeth filtering: record only the difference from the neighboring pixel values, since value of pixels changes gradually in most cases

Phases for Simple Image Compression

- Phase 1: Apply a simplified Paeth filtering algorithm to the grayscale image to reduce the range of pixel values
- Phase 2: Encode those values in a more compact binary representation

Phase I: Simplified Paeth Filtering

- Find the average of three neighboring pixels in left, upper, and upper-left positions
 - When neighboring pixel doesn't exist, exclude it from the calculation



Input Image S

$$\text{Avg}[0][0] = 0$$

$$\text{Avg}[0][1] = (S[0][0]) / 1 = A / 1$$

$$\text{Avg}[0][2] = (S[0][1]) / 1 = B / 1$$

$$\text{Avg}[1][0] = (S[0][0]) / 1 = A / 1$$

$$\text{Avg}[1][1] = (S[1][0] + S[0][1] + S[0][0]) / 3 = (D + B + A) / 3$$

Phase I: Simplified Paeth Filtering

2. Get the filtered value by computing the difference between pixel value and average value
 - To prevent getting a negative value, if pixel value is smaller than average value, add 256 before subtracting

```
Filter[i][j] = S[i][j] - Avg[i][j]           if S[i][j] >= Avg[i][j]
                S[i][j] + 256 - Avg[i][j]     otherwise
```

Example: Phase I

0	0	0	0
50	75	100	120
75	100	120	0

Input Image S

$$\text{Avg}[0][0] = 0$$

$$\text{Avg}[0][1] = 0/1 = 0$$

$$\text{Avg}[0][2] = 0/1 = 0$$

$$\text{Avg}[0][3] = 0/1 = 0$$

$$\text{Avg}[1][0] = 0/1 = 0$$

$$\text{Avg}[1][1] = (50 + 0 + 0)/3 = 16$$

$$\text{Avg}[1][2] = (75 + 0 + 0)/3 = 25$$

$$\text{Avg}[1][3] = (100 + 0 + 0)/3 = 33$$

$$\text{Avg}[2][0] = 50/1 = 50$$

$$\text{Avg}[2][1] = (75 + 75 + 50)/3 = 66$$

$$\text{Avg}[2][2] = (100 + 100 + 75)/3 = 91$$

$$\text{Avg}[2][3] = (120 + 120 + 100)/3 = 113$$

$$\text{Filter}[0][0] = 0 - 0 = 0$$

$$\text{Filter}[0][1] = 0 - 0 = 0$$

$$\text{Filter}[0][2] = 0 - 0 = 0$$

$$\text{Filter}[0][3] = 0 - 0 = 0$$

$$\text{Filter}[1][0] = 50 - 0 = 50$$

$$\text{Filter}[1][1] = 75 - 16 = 59$$

$$\text{Filter}[1][2] = 100 - 25 = 75$$

$$\text{Filter}[1][3] = 120 - 33 = 87$$

$$\text{Filter}[2][0] = 75 - 50 = 25$$

$$\text{Filter}[2][1] = 100 - 66 = 34$$

$$\text{Filter}[2][2] = 120 - 91 = 29$$

~~$$\text{Filter}[2][3] = 0 - 113 = -113$$~~

$$S[2][3] < \text{Avg}[2][3]$$

$$\text{Filter}[2][3] = 0 + 256 - 113 = 143$$

$$113 + 143 = 256 = \underbrace{100000000}_{0}^{(2)}$$

Phase 2: Encoding Filtered Values

- Use minimum filtered value as base value of the row & calculate the deltas from the base value

0	0	0	0
50	75	100	120
75	100	120	0

Input Image S[3][4]

0	0	0	0
50	59	75	87
25	34	29	143

Filter[3][4]

0	0	0	0
0	9	25	37
0	9	4	118

base(0): 0

base(1): 50

base(2): 25

Delta[3][4]

Phase 2: Encoding Filtered Values

4. Find the number of bits needed for representing the delta

- Can be calculated from the maximum delta value for each row

```
n(i) = 0      if max(Delta[i]) == 0,  
1           else if max(Delta[i]) == 1,  
2           else if max(Delta[i]) < 4,  
3           else if max(Delta[i]) < 8,  
4           else if max(Delta[i]) < 16,  
5           else if max(Delta[i]) < 32,  
6           else if max(Delta[i]) < 64,  
7           else if max(Delta[i]) < 128,  
8           otherwise
```

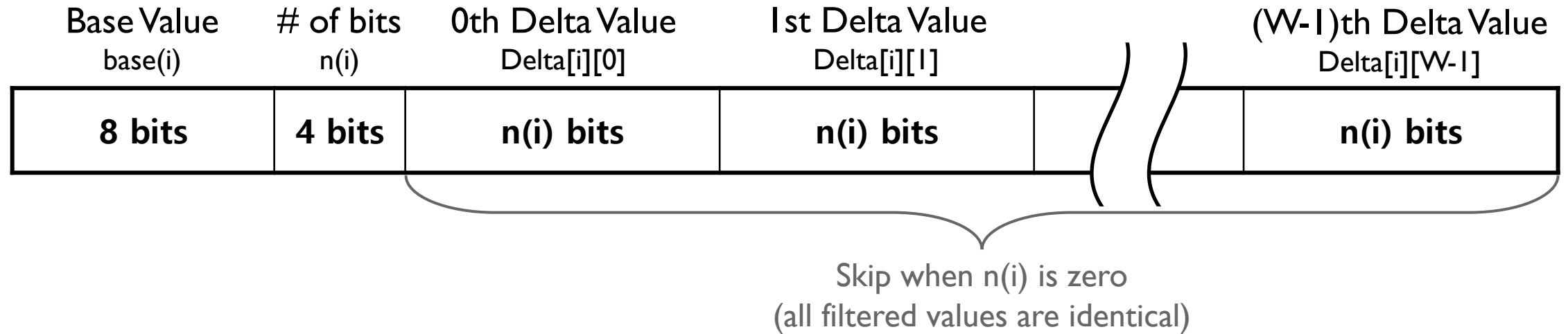
of bits $n(i)$ needs to encode delta values with unsigned integers

0	0	0	0	$n(0) = 0$
0	9	25	37	$n(1) = 6$
0	9	4	118	$n(2) = 7$

Delta[3][4]

Phase 2: Encoding Filtered Values

5. Now encode each row at a time using the format below



Delta[3][4]

0	0	0	0
0	9	25	37
0	9	4	118

$base(0) = 0, n(0) = 0$

$base(1) = 50, n(1) = 6$

$base(2) = 25, n(2) = 7$

$base(i)$ $n(i)$

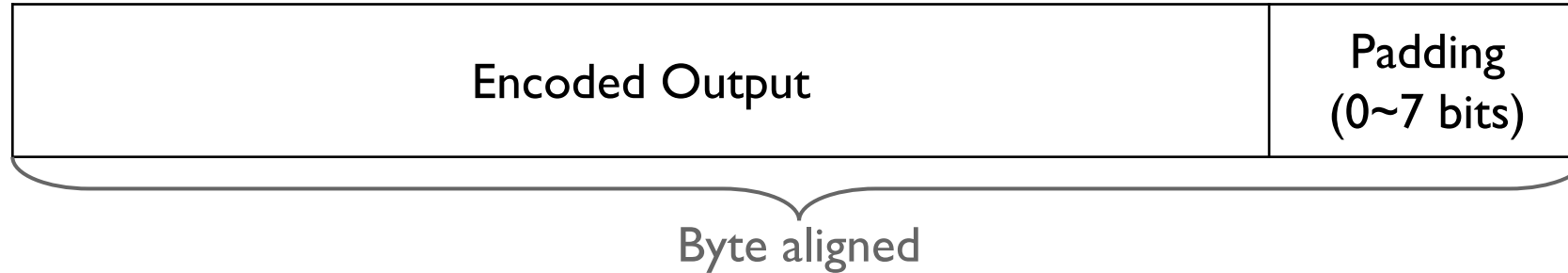
Row 0: 00000000 0000

Row 1: 00110010 0110 000000 001001 011001 100101

Row 2: 00011001 0111 0000000 0001001 0000100 1110110

Phase 2: Encoding Filtered Values

6. If the total number of output bits is not a multiple of 8, pad 0's until it becomes a multiple of 8



	base(i)	n(i)				
Row 0:	00000000	0000				
Row 1:	00110010	0110	000000	001001	011001	100101
Row 2:	00011001	0111	0000000	0001001	0000100	1110110

Output: 00000000 00000011 00100110 00000000 10010110 01100101 00011001 01110000 00000010 01000010 01110110

0x00 0x02 0x26 0x00 0x96 0x65 0x19 0x70 0x02 0x42 0x76

Example (2)

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0


Input Image S[5][10]

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Filter[5][10]

	base(i)	n(i)
Row 0:	00000000	0000
Row 1:	00000000	0000
Row 2:	00000000	0000
Row 3:	00000000	0000
Row 4:	00000000	0000

Output: $\begin{matrix} \text{row 0} & \text{row 1} & \text{row 2} & \text{row 3} \\ 00000000 & 00000000 & 00000000 & 00000000 \\ \text{row 4} \\ 00000000 & 00000000 & = 0x00 & 0x00 & 0x00 & 0x00 & 0x00 & 0x00 & 0x00 & 0x00 \end{matrix}$



Specification

- All you need to do is to write `encode()` function given in `pal.c`
- `int encode(const u8* src, const int width, const int height, u8* dst);`
`{ /* fill this function */ }`
 - `src` points to the memory address of the input data
 - `width` and `height` are the width & height of input data (in bytes)
 - `dst` points to the memory address for encoded result
 - It returns the length of the output (in bytes)
 - If `width` or `height` is zero, return zero

Restrictions

- Contents of the buffer after the encoded output should not be corrupted

dst buffer for example (2): 11111111 11111111 11111111 11111111 11111111
11111111 11111111 11111111 11111111 11111111

Output for example (2): 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00111111 11111111

Contents after the encoded output is corrupted!

- You are not allowed to use any array
- You are not allowed to use any library functions
- Your solution should finish within a reasonable time

Submission

- **Due: 11:59PM, September 18 (Sunday)**
 - 25% of the credit will be deducted for every single day delay
- **Only submit the pa1.c file to the submission server**
 - You don't have to write a report in this assignment

Slip Days

- You can use up to 4 slip days during this semester
 - To use slip days, please post how many slip days you want to use on QnA board
- We highly recommend to save slip days for next projects!

SNU: Systems Software & Architecture Lab.

CLASS LIST

[4190.308] Computer Architecture

CLASS HOME

Overview

Projects

Rank

Latest Runs

QnA

Course Homepage

SYSTEM

Test Queue Status

Member Info

Logout

4190.308 : Q n A

#	Subject	Writer	Date	Hit
<input type="text"/>				<input type="button" value="Search"/>
				<input type="button" value="Write"/>
				« 1 »

<Scoring ratio for last semester>

Projects 40%

Project #1 5%

Project #2 8%

Project #3 13%

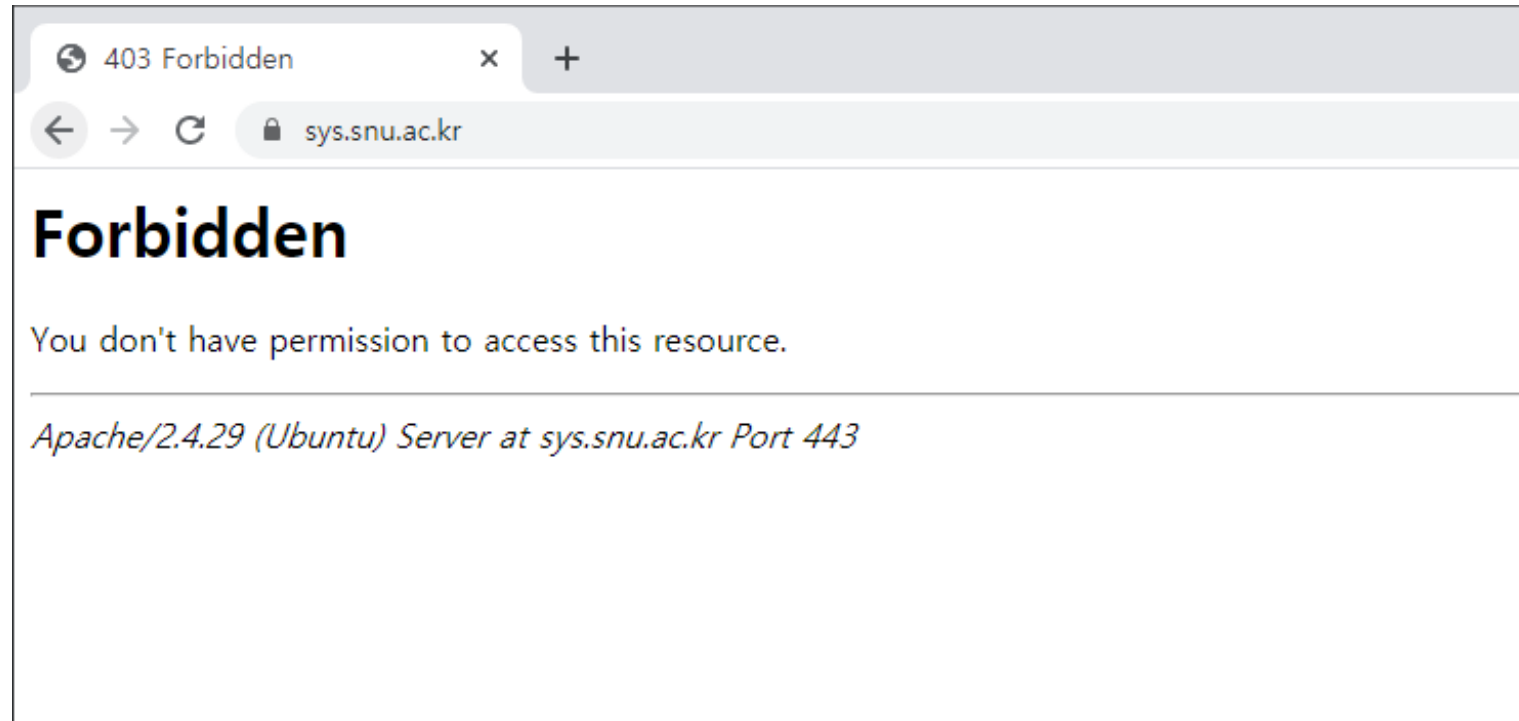
Project #4 14%

✘ It is not for this semester.

Account Registration

How to Access the Submission Server

- <http://sys.snu.ac.kr>
- Need to access via in-school IP or authorized IP
 - In-school IP: 147.46.X.X or 147.47.X.X



How to Get My IP Authorized?

- Please submit your IP through Google Form (<https://forms.gle/rbWd2ZV2mAxRTIAr5>)
- To get your IP address, search “what is my ip” in Google
 - Note that virtual IP addresses are not valid
 - 10.0.0.0 ~ 10.255.255.255
 - 172.16.0.0 ~ 172.31.255.255
 - 192.168.0.0 ~ 192.168.255.255

How to Register an Account?

- You should make an account to submit your assignments
- Please make an account with correct name and student number

SNU: Systems Software & Architecture Lab.

Login

Email

Password

We recommend the Google Chrome web browser.
© SNU Systems Software & Architecture Laboratory

SNU: Systems Software & Architecture Lab.

Join

Email

Password

Password Confirm

Name

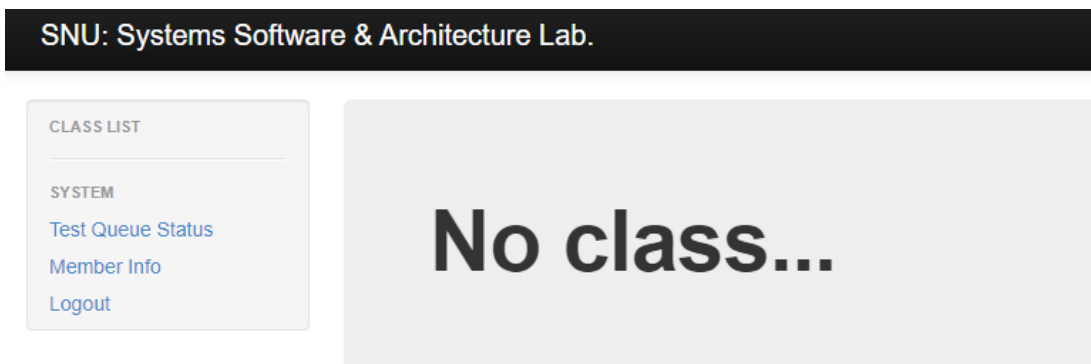
NickName

Student No.

Mobile - -

How to Register an Account?

- Don't worry if there's "No class" on the screen
- We will join you to class as soon as possible



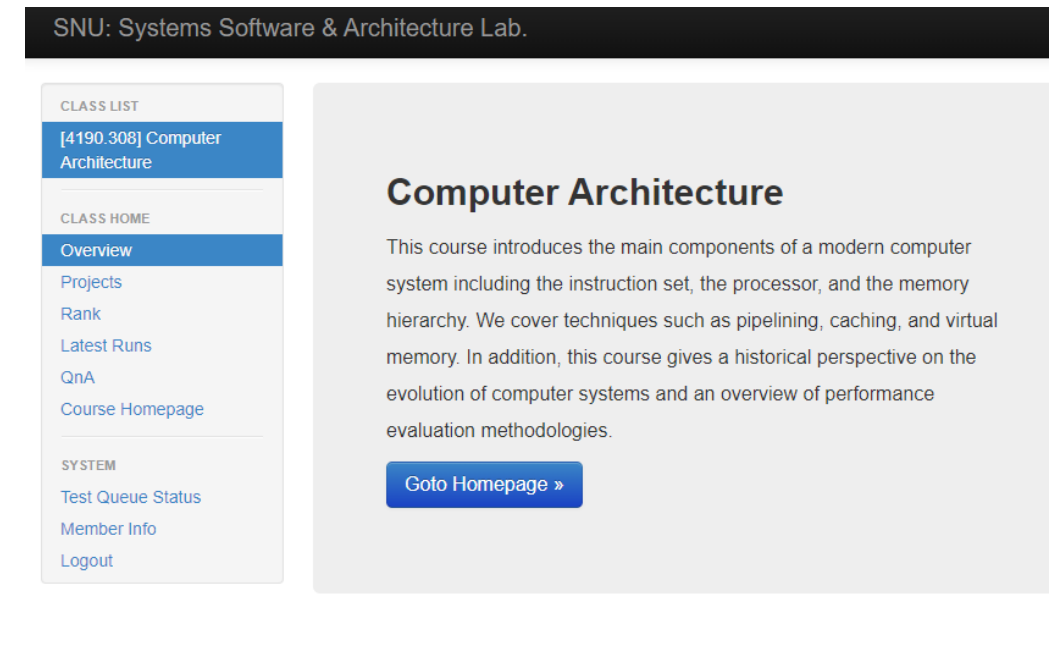
SNU: Systems Software & Architecture Lab.

CLASS LIST

SYSTEM

- [Test Queue Status](#)
- [Member Info](#)
- [Logout](#)

No class...



SNU: Systems Software & Architecture Lab.

CLASS LIST

- [4190.308] Computer Architecture**

CLASS HOME

- Overview**
- [Projects](#)
- [Rank](#)
- [Latest Runs](#)
- [QnA](#)
- [Course Homepage](#)

SYSTEM

- [Test Queue Status](#)
- [Member Info](#)
- [Logout](#)

Computer Architecture

This course introduces the main components of a modern computer system including the instruction set, the processor, and the memory hierarchy. We cover techniques such as pipelining, caching, and virtual memory. In addition, this course gives a historical perspective on the evolution of computer systems and an overview of performance evaluation methodologies.

[Goto Homepage »](#)

We recommend the Google Chrome web browser.

© SNU Systems Software & Architecture Laboratory

Thank You!

- Don't forget to read detailed description before you start your assignment
- If you have any question about the assignment, feel free to ask via email or eTL
- This file will be uploaded after the lab session 😊