

Microprocessor Application

Project - ARM Code Optimization for Image Converting
with Keil MDK tool

Young Seo Lee
(yslee@ssu.ac.kr)

Intelligent System and Architecture Computing (ISAC) Lab.



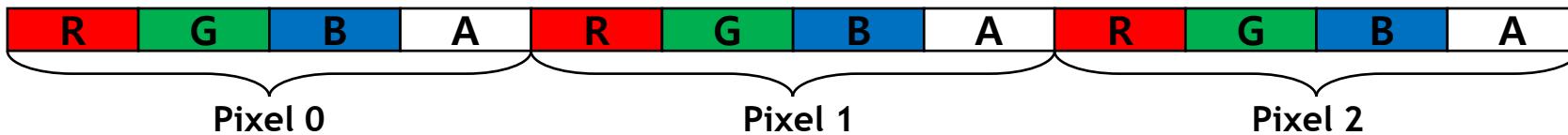
Image Processing



Pixel
(Red, Green, Blue, Alpha)

960x640 image size
= 614,400 pixels
= 2,457,600 bytes
(= 614,400 * 4 bytes)

- 이미지는 다수의 Pixel로 구성 → 이미지 크기만큼의 pixel 개수 포함
- 하나의 Pixel 값은 다양한 정보를 포함하고 있음
 - Red, Green, Blue, Alpha (투명도) 정보를 포함하여 보통 RGBA format을 가진다고 함
 - 일반적인 RGBA format은 각 구성요소 당 8-bit (=1-byte)로 이루어져 있음
 - 1 Pixel (32-bit, 4-byte) = Red (8-bit) + Green (8-bit) + Blue (8-bit) + Alpha (8-bit)
- 메모리에는 Pixel 당 데이터가 저장되어 있음



Subproject #1: Red Pixel Count



Pixel
(Red, Green, Blue, Alpha)

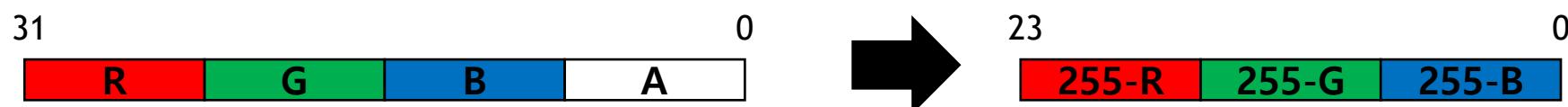
- RGBA image 중에서 “Red” 부분이 128 이상인 pixel의 개수를 세기
 - Red 부분은 8-bit로 구성 (integer 가정 시 0~255 범위 표현 가능)
- 최종 결과는 “개수” (number)



Subproject #2: Image Negative (색 반전)



- RGBA image 중에서 Red, Green, Blue 각각의 정보를 반전
 - 각각 8-bit로 구성되어 있으므로 255 -(기존 값) 연산을 하면 쉽게 반전 가능



Subproject #3: Grayscale



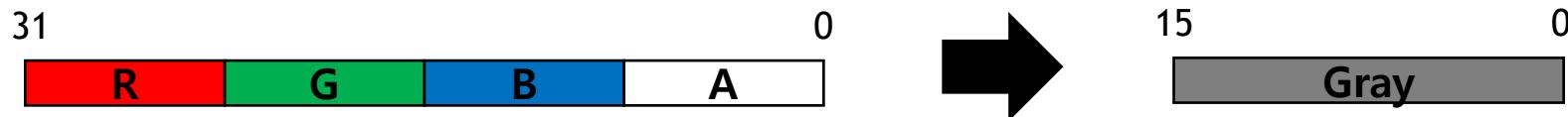
<RGBA image>



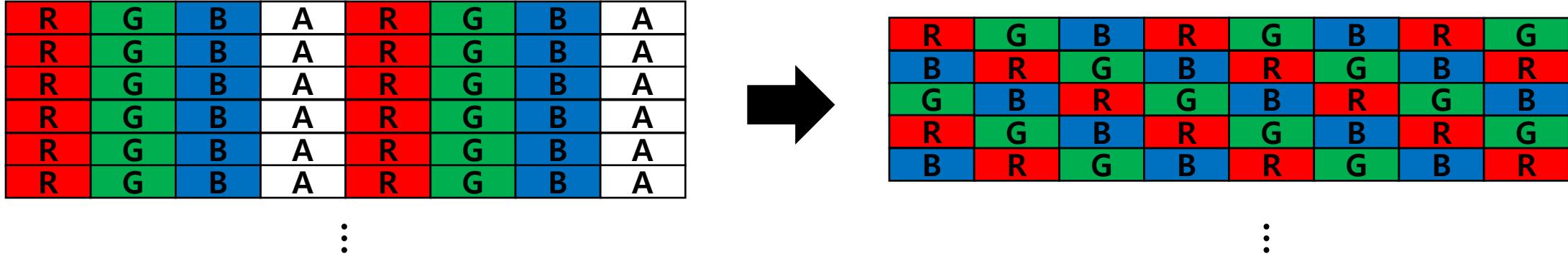
<Grayscale image>

Pixel = 16-bit
(grayscale)

- RGBA image는 사람이 잘 보기 위한 목적이지, 사물 인식 등 이미지 활용 분야에서는 적합하지 않음
- 특정 분야에서는 원활한 활용을 위해 Grayscale 이미지로 변환
 - Grayscale (16-bit) 변환 방법 : $\text{Gray} = 3 * R + 6 * G + 1 * B$
- 주의 사항: 결과 값을 16-bit data로 만들어 주어야 함
(예를 들어, 결과 값이 10-bit라면 16-bit로 zero padding 필요)
- Grayscale 이미지는 Pixel 값 당 16-bit의 명암 (grayscale) 정보만을 가지고 있음

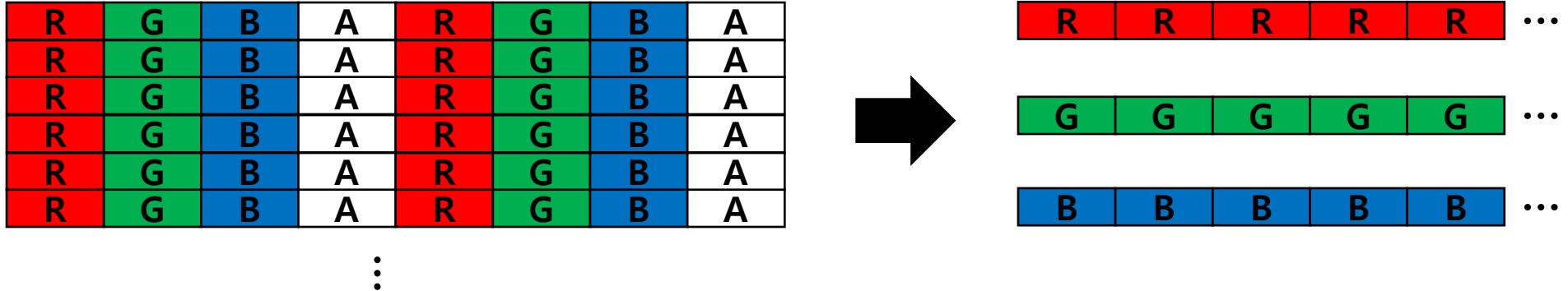


Memory relocation



- 위 3개의 Image Converting 과정에서 RGBA format 중 A (alpha)는 연산에 활용되지 않음
- 이미지를 메모리에 탑재하는 과정에서 A를 제외하고 RGB만 배치하면 어떤 이슈가 생길까?

Memory relocation



- 위 3개의 Image Converting 과정에서 RGBA format 중 A (alpha)는 연산에 활용되지 않음
- 이미지를 메모리에 탑재하는 과정에서 A를 제외하고 **RGB를 각각 따로 배치**
- Image converting 함수들에 대해서 ARM code 추가 최적화 가능 → 왜?

Project Summary

- Image processing 및 image converting 과정의 이해
- Keil MDK를 활용해 Image converting을 수행하는 ARM code 작성
 - 기존 이미지의 특정 Red pixel의 개수를 세는 ARM code 작성
 - 기존 이미지를 24-bit 색 반전 이미지로 변환하는 ARM code 작성
 - 기존 이미지를 16-bit grayscale 이미지로 변환하는 ARM code 작성
- Image converting을 수행하는 ARM code 최적화
- 메모리 재배치를 통한 이미지 변환 시 ARM code 최적화
- 기존 ARM code와 최적화된 ARM code 사이의 성능 비교 분석

Project Summary

■ Image binary (hex) 파일 제공

- 프로젝트 실행을 위해서는 image 파일을 hex 파일로 변환하는 과정이 필요
- 수행할 필요 없이, 프로젝트 실행을 위한 hex 파일을 제공

```
1 #include <stdio.h>
2 #include <stdint.h>
3
4 #define BUFFER_SIZE 8
5
6 int main(int argc, char *argv[]) {
7     if (argc != 2) {
8         printf("Usage: %s <filename>\n", argv[0]);
9         return 1;
10    }
11
12    char *filename = argv[1];
13    uint8_t buffer[BUFFER_SIZE];
14    size_t bytes_read;
15
16    FILE *file = fopen(filename, "rb");
17    if (!file) {
18        printf("Error opening file.\n");
19        return 1;
20    }
21
22    printf("Binary data for file: %s\n", filename);
23
24    while ((bytes_read = fread(buffer, sizeof(uint8_t), BUFFER_SIZE, file)) > 0) {
25        for (size_t i = 0; i < bytes_read; i++) {
26            printf("%02X ", buffer[i]); // 각 바이트를 16진수 형태로 출력
27        }
28        printf("\n");
29    }
30
31    fclose(file);
32    return 0;
33 }
```

Image file to binary(hex) code
(simple example)



89	50	4E	47	0D	0A	1A	0A
00	00	00	0D	49	48	44	52
00	00	03	C0	00	00	02	80
08	06	00	00	00	53	F9	11
43	00	00	00	01	73	52	47
42	00	AE	CE	1C	E9	00	00
20	00	49	44	41	54	78	5E
5C	BD	77	73	23	59	92	EC
1B	29	21	08	B2	44	77	8F
58	71	67	EF	B5	77	FF	7B
DF	FF	F3	3C	B3	B5	DD	9D
6D	59	82	02	40	AA	67	3F
F7	08	80	B3	1C	EB	A9	2A
92	00	32	4F	9E	13	C2	C3

Project Summary

▪ Image binary (hex) 파일 분석

89	50	4E	47	0D	0A	1A	0A
00	00	00	0D	49	48	44	52
00	00	03	C0	00	00	02	80
08	06	00	00	00	53	F9	11
43	00	00	00	01	73	52	47
42	00	AE	CE	1C	E9	00	00
20	00	49	44	41	54	78	5E
5C	BD	77	73	23	59	92	EC
1B	29	21	08	B2	44	77	8F
58	71	67	EF	B5	77	FF	7B
DE	FF	F3	3C	B3	B5	DD	9D
6D	59	82	02	40	AA	67	3F
F7	08	80	B3	1C	EB	A9	2A
92	00	32	4F	9E	13	C2	C3

File header info.

Width = 0x03C0 (960)

Height = 0x0280 (640)

IDAT → Pixel start

Pixel 0 = 0x 78 5E 5C BD

Pixel 1 = 0x 77 73 23 59

Pixel 2 = 0x 92 EC 1B 29

...

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	 	Space	64	40	100	@	@
1	1	001	Start of Header	33	21	041	!	!	65	41	101	A	A
2	2	002	Start of Text	34	22	042	"	"	66	42	102	B	B
3	3	003	End of Text	35	23	043	#	#	67	43	103	C	C
4	4	004	End of Transmission	36	24	044	$	\$	68	44	104	D	D
5	5	005	Enquiry	37	25	045	%	%	69	45	105	E	E
6	6	006	Acknowledgment	38	26	046	&	&	70	46	106	F	F
7	7	007	Bell	39	27	047	'	'	71	47	107	G	G
8	8	010	Backspace	40	28	050	((72	48	110	H	H
9	9	011	Horizontal Tab	41	29	051))	73	49	111	I	I
10	A	012	Line feed	42	2A	052	*	*	74	4A	112	J	J
11	B	013	Vertical Tab	43	2B	053	+	+	75	4B	113	K	K
12	C	014	Form feed	44	2C	054	,	,	76	4C	114	L	L
13	D	015	Carriage return	45	2D	055	-	-	77	4D	115	M	M
14	E	016	Shift Out	46	2E	056	.	.	78	4E	116	N	N
15	F	017	Shift In	47	2F	057	/	/	79	4F	117	O	O
16	10	020	Data Link Escape	48	30	060	0	0	80	50	120	P	P
17	11	021	Device Control 1	49	31	061	1	1	81	51	121	Q	Q
18	12	022	Device Control 2	50	32	062	2	2	82	52	122	R	R
19	13	023	Device Control 3	51	33	063	3	3	83	53	123	S	S
20	14	024	Device Control 4	52	34	064	4	4	84	54	124	T	T
21	15	025	Negative Ack.	53	35	065	5	5	85	55	125	U	U
22	16	026	Synchronous idle	54	36	066	6	6	86	56	126	V	V
23	17	027	End of Trans. Block	55	37	067	7	7	87	57	127	W	W
24	18	030	Cancel	56	38	070	8	8	88	58	130	X	X
25	19	031	End of Medium	57	39	071	9	9	89	59	131	Y	Y
26	1A	032	Substitute	58	3A	072	:	:	90	5A	132	Z	Z
27	1B	033	Escape	59	3B	073	;	;	91	5B	133	[{
28	1C	034	File Separator	60	3C	074	<	<	92	5C	134	\	\
29	1D	035	Group Separator	61	3D	075	=	=	93	5D	135]	}
30	1E	036	Record Separator	62	3E	076	>	>	94	5E	136	^	^
31	1F	037	Unit Separator	63	3F	077	?	?	95	5F	137	_	_

asciicharstable.com

Project Summary

■ 프로젝트 수행 시 유의사항

- 이미지의 pixel이 커서 메모리 공간 부족 발생 가능성 있음
- Image binary (hex) 파일에 대해서 전부가 아닌 일부에 대해서만 수행
- **전부가 아닌 9,600 pixels에 대해서만 실험 수행**
 - 각 pixel 당 32-bit (RGBA, 4-byte) 데이터 저장
 - 전체 pixel의 수는 $960(\text{가로}) * 480(\text{세로}) = 460,800 \text{ pixels}$ (전부 수행 X)
 - $960(\text{가로}) * 10(\text{세로}) = 9,600 \text{ pixels}$ 에 대해서만 실험 수행
 - = 연속적인 9,600 pixels

How to Score the Project

■ 제출 기한 및 내용

- 프로젝트는 각 팀에서 1명 (e.g., 팀장)이 LMS를 통해 대표 제출
- **기한: 6월 8일(일) 23:59 (지각 제출은 받지 않음 = 0점)**
- **내용: Project_[Group_number].zip** (압축) (e.g., Project_1팀.zip)*
→ **프로젝트 전체 압축 파일은 아래 파일들을 반드시 포함해야 함**
 - ① **보고서:** Report_[Group_number].pdf (e.g., Report_1팀.pdf)
 - ② **발표자료:** Presentation_[Group_number].ppt (e.g., Presentation_1팀.pdf)
 - ③ **발표녹화:** Record_[Group_number].mov (e.g., Record_1팀.mov)
 - 동영상 파일의 확장자는 정해진 forma이 있는 것은 아님 (다른 확장자도 상관 없음)
- **제출 기한 및 내용과 포맷이 올바르지 않을 경우 점수를 부여하지 않음**

* 본인의 팀이 몇 번인지는 LMS 공지를 참고하기 바랍니다

How to Score the Project

■ 개별 기여도 제출 (**개인 제출**)

- [별첨 2]의 format을 확인하여 팀원의 개별 기여도를 LMS를 통해 제출
 - [별첨 2] 표에 직접 기입 후 오른쪽 버튼 클릭 → [그림으로 저장]
- 기한: 6월 8일(일) 23:59 (**프로젝트 제출일과 동일**)
- 내용
 - 포맷: Project_Score_[Group_number]_[name].jpg (e.g., Project_Score_1팀_홍길동.jpg)



* 본인의 팀이 몇 번인지는 LMS 공지를 참고하기 바랍니다

■ 보고서는 아래 내용을 포함해야 함

- ① 프로젝트의 개요
- ② 32-bit RGBA 이미지를 메모리의 연속된 주소공간에 저장하는 함수의 ARM code
- ③ 3가지 이미지 변환 함수의 동작 방식 및 block diagram (순서도)
 - Function #1: 32-bit RGBA image → pixel number
 - Function #2: 32-bit RGBA image → 24-bit RGB image (negative)
 - Function #3: 32-bit RGBA image → 16-bit grayscale image
- ④ 3가지 이미지 변환 함수의 C언어 코드 혹은 ARM code
 - 각 Function 별 C언어 코드 혹은 ARM code 결과물
 - 각 Function 별 시뮬레이션 결과 및 설명 (함수 수행 전과 후의 memory map 정보 포함)
- ⑤ ARM code 최적화를 위한 C언어 코드 및 ARM code
 - Memory relocation (32-bit RGBA pixels → {8-bit R}, {8-bit G}, {8-bit B})
 - RGBA에서 A (alpha)만 제외하고 24-bit RGB를 R, G, B로 따로 저장했을 때 어떤 특이점이 생기는지 서술
 - Relocation 이후 변경된 Function #1, #2, #3의 코드 (e.g., {8-bit R}, {8-bit G}, {8-bit B}) → 16-bit grayscale image)
 - ④와 마찬가지로 각 Function 별 시뮬레이션 결과 및 설명 (함수 수행 전과 후의 memory map 정보 포함)
 - ④의 결과와 일치하는 결과가 나오는지 검증

■ 보고서는 아래 내용을 포함해야 함 (계속)

⑥ (가산점) ARM code 최적화

- ARM 명령어 변경을 통한 추가 최적화 달성 (e.g., 총 명령어 개수 감소 등)
- 추가 최적화 달성 시 기존 결과와 동일한 결과가 나오는지 검증 (debugging 결과 반드시 포함)

⑦ ④의 ARM code와 ⑤의 ARM code 사이의 성능 비교

- 각 이미지 변환 함수를 위한 전체 수행 시간, 필요 명령어 개수 등 비교

⑧ 위 모든 과정의 결과물

⑨ 프로젝트 수행일지 ([별첨 1]의 format 확인 필수)

⑩ Troubleshooting

- 프로젝트 진행 과정에서 발생한 문제 및 어떻게 해결 하였는지에 대한 상세한 설명

Presentation and Recording Summary

■ 발표자료

- 보고서의 내용을 요약해서 발표자료로 구성 (별도 ppt 파일로 작성)
- C언어 코드 및 ARM code는 자세히 포함될 필요는 없음 (보고서에서 자세히 기술하면 됨)
- 발표자료의 분량 제한은 별도로 없음

■ 발표녹화

- 발표녹화는 10~15분 분량으로 구성 (초과해도 됨)
- 녹화 시 화면 및 음성을 녹화할 수 있는 프로그램 활용
 - 프로그램 예시: 데모크리에이터 (<https://dc.wondershare.kr/>), 반디캠 (<https://www.bandicam.co.kr/>) 등
 - MacOS 사용 시 내장된 Quicktime 등 활용 가능

[별첨 1] 프로젝트 수행 일지

이름	조원1(조장)	조원2	조원3	조원4	조원5
모임일자	수행내용				
5/8	(예시) - 프로젝트의 개요 및 목표에 대한 논의 - 각자의 진행 방향에 대한 역할 분담 1) 조원1: ~~~ 2) 조원2: ~~~				
5/15	-	-	-		
5/22	-	-	-		
5/29	(예시) - 기준 코드와 최적화된 코드 검증 과정에서 에러 발생 - ~~~ 에러가 발생했고 ~~~ 해결				
...	...				

[별첨 2] 개별 기여도 평가 (개인 제출)

이름	조원1	조원2	조원3	조원4 (본인)	조원5
기여도 점수 (최대 4점, 중복X)	4	3	2	본인은 평가 X	1
역할 (최대한 자세히 서술)	(예시) - Project 관리 - ~~~ 부분 코드 작성 - 관련 그림 - 보고서 XX% 작성	(예시) - 발표자료 XX% 작성 - 코드 디버깅	(예시) - 보고서 XX% 작성		
본인 제외 최대 코드 작성자	O				
본인 제외 최대 보고서 작성자	O				
본인 제외 최대 발표자료 및 녹화자료 기여자		O			

- **기여도 점수: 본인을 제외하고 나머지 조원에 대한 총 평가 작성**

- 5인 팀: 1등 (4점), 2등 (3점), 3등 (2점), 4등 (1점), 본인 (제외)
- 4인 팀: 1등 (4점), 2등 (3점), 3등 (2점), 본인 (제외)
- 3인 팀: 1등 (4점), 2등 (3점), 본인 (제외)

- **본인 제외 ~~ 작성자(기여자)**

- 본인 제외 해당 부분에 최대 기여하였다고 생각하는 사람에게 표시
- 해당하는 사람이 없다고 판단될 경우에는 체크하지 않아도 됨

Questions?



E-mail: yslee@ssu.ac.kr

Lab: <https://sites.google.com/view/isac-ssu>

