CS 2110 Quiz 6

Izaan Kamal

TOTAL POINTS

22.5 / 100

QUESTION 1

What's the Point? 24 pts

1.1 shortValue 0 / 5

✓ **+ 0 pts Graded**

+ 5 pts Correct: $$\texttt{*(b + 6)}$$ or equivalent

Note: No credit for use of $$\texttt{[]}$$ and $$\texttt{]}$$

+ 2.5 pts Partial: Off-by-one

i.e. $$\texttt{*(b + 7)}$$

1.2 intAddr 0 / 5

✓ **+ 0 pts Graded**

+ 5 pts Correct: $$\texttt{(c + 1)}$$ or equivalent

Note: No credit for use of $$\texttt{[]}$$ and $$\texttt{]}$$

+ 2.5 pts Partial: Off-by-one

i.e. $$\texttt{(c + 2)}$$

1.3 charValue 0 / 7

✓ **+ 0 pts Graded**

+ 7 pts Correct:

$$\texttt{*(*(a + 3) + 2)}$$ or $$\texttt{*((char*) a + 32)}$$

... or equivalent

Note: No credit for use of $$\texttt{[]}$$ and $$\texttt{]}$$.

Note: Not casting $$\texttt{a}$$ to $$\texttt{(char*)}$$ gives a different address, since $$\texttt{a}$$ decays to a pointer to $$\texttt{array of 10 char}$$ (i.e. $$\texttt{sizeof(*a) == 10}$$)

+ 3.5 pts Partial credit:

Didn't recognize $$\texttt{sizeof(*a) == 10}$$

e.g. $$\texttt{*(a + 3 * (10) + 2)}$$

1.4 structAddr 0 / 7

✓ **+ 0 pts Graded**

+ 7 pts Correct:

$$\texttt{5 * 20 * 30 + 7 * 30 + 10 (i.e. 3220)}$$

...or equivalent

Note: No credit for use of $$\texttt{[]}$$ and $$\texttt{]}$$

+ 5 pts Partial credit:

Did not use $$\texttt{pd}$$ (used $$\texttt{d}$$ instead) but cast it correctly and calculated the correct index

Note: No credit for use of $$\texttt{[]}$$ and $$\texttt{]}$$

+ 3.5 pts Partial credit:

Index calculation was off-by-one: $$\texttt{3219}$$ or $$\texttt{3221}$$

+ 3.5 pts Partial credit:

Small, careless typo (i.e. one wrong character)

+ 3.5 pts Partial credit:

Index calculation was correct but multiplied by $$\texttt{sizeof(struct s)}$$

QUESTION 2

2 Searching for a Book 5 / 16

✓ + 0 pts Graded

✓ + 5 pts **Correct return type:**

$$\texttt{Book *}$$ or $$\texttt{void*}$$

Note: No credit for $$\texttt{const Book*}$$ or $$\texttt{const void*}$$

Note: Syntax errors or compilation problems related to improperly declared return values are absorbed by this criteria (e.g. $$\texttt{*Book}$$).

+ **6 pts** Correct function pointer name:

$$\texttt{(*bookComp)(...)}$$

Must properly declare as a function pointer with the correct name.

Note: No credit if the full parameter does not compile, excluding cases described in other rubric criteria.

+ **5 pts** Correct parameter types:

$$\texttt{(Book*, Book*)}$$ or
$$\texttt{(const Book*, const Book*)}$$ or
$$\texttt{(void*, void*)}$$ or
$$\texttt{(const void*, const void*)}$$

Note: Syntax errors or compilation problems related to parameters are absorbed by this criteria

(e.g. $$\texttt{...(*bookComp)(*Book, *Book)}$$ or $$\texttt{...(*bookComp)((Book*), (Book*))}$$).

+ **0 pts** Sample correct answers:

$$\texttt{Book* (*bookComp)(Book*, Book*)}$$ or $$\texttt{Book* (*bookComp)(const Book*, const Book*)}$$

QUESTION 3

3 Extracting Channels of a Pixel 7.5 / 20

✓ + 0 pts Graded

+ **7.5 pts** Returns through parameters correctly:

$$\texttt{*red \: \: = ... ;}$$
$$\texttt{*green = ... ;}$$
$$\texttt{*blue \: = ... ;}$$

✓ + 5 pts **Uses correct masks and operation ($$\texttt{\&}$$) for each channel:**

$$1023_{10}$$ or $$\texttt{0x3FF}$$ or equivalent

Must get all three correct.

Note: Binary literals will not compile with the specified flags.

Note: Cannot assume bits $$\texttt{31}$$ and $$\texttt{30}$$ are zero. Must mask them out of green channel to receive credit.

+ **5 pts** Shifts $$\text{pixel}$$ correctly for each channel:

Red ($$\texttt{>> 10}$$), green ($$\texttt{>> 20}$$), and blue ($$\texttt{>> 0}$$)

Must get all three correct.

Note: The channels must be returned through the lower ten bits of the color parameters

Note: Cannot assume bits bits $$\texttt{31}$$ and $$\texttt{30}$$ are zero.

✓ **+ 2.5 pts No small syntax errors:**

Missing semicolon, uses $$\texttt{\&\&}$$ instead of $$\texttt{\&}$$, $$\texttt{x3FF}$$, etc.

QUESTION 4

Drawing a Collage with DMA 40 pts

4.1 Loop condition 10 / 10

+ **0 pts** Graded

✓ **+ 10 pts Correct: $$\texttt{row < width}$$**

+ **7.5 pts** Partial (minor iteration error):

$$\texttt{row <= width}$$ or
$$\texttt{row < width - 1}$$ or
$$\texttt{row < width + 1}$$

+ **7.5 pts** Partial (syntax error):

Used semicolon

+ **5 pts** Partial (variable name error):

$$\texttt{wrong\_var\_name < width}$$ or
$$\texttt{row < wrong\_var\_name}$$

Note: $$\texttt{wrong\_var\_name}$$ must be something like $$\texttt{i}$$, $$\texttt{r}$$, $$\texttt{image -> width}$$, etc. Not simply any wrong variable.

4.2 Image portion: SRC 0 / 5

✓ **+ 0 pts Graded**

+ **5 pts** Correctly specifies the $$\texttt{.src}$$:

$$\texttt{image + OFFSET(row, 0, width)}$$

i.e. $$\texttt{image + row * width}$$

Note: Could have also used decrementing logic

+ **2.5 pts** Partial credit for the $$\texttt{.src}$$: Calculates the correct offset but passes a $$\texttt{u16}$$ instead of $$\texttt{u16*}$$

i.e. $$\texttt{image[OFFSET(row, 0, width)]}$$

- **1.5 pts** Deduction: Correct with minor var. name error

i.e. $$\texttt{img}$$, $$\texttt{r}$$, etc.

4.3 Image portion: DST 0 / 5

✓ **+ 0 pts Graded**

+ **5 pts** Correctly specifies the $$\texttt{.dst}$$:

$$\texttt{videoBuffer + OFFSET(row, 0, GBA\_WIDTH)}$$

i.e. $$\texttt{videoBuffer + (row * GBA\_WIDTH)}$$

Note: Could have also used decrementing logic

+ **2.5 pts** Partial credit for the $$\texttt{.dst}$$: Calculates the correct offset but passes a $$\texttt{u16}$$ instead of $$\texttt{u16*}$$

i.e. $$\texttt{videoBuffer[OFFSET(row, 0, GBA\_WIDTH)]}$$

- **1.5 pts** Deduction: Correct with minor var. name error

i.e. $$\texttt{img}$$, $$\texttt{r}$$, etc.

4.4 Image portion: CNT 0 / 5

✓ **+ 0 pts Graded**

+ **3 pts** Correctly specifies the $$\texttt{.cnt}$$ flags:

$$\texttt{\: \: - DMA\_ON}$$
$$\texttt{\: \: - DMA\_DST\_INC}$$
$$\texttt{\: \: - DMA\_SRC\_INC}$$

Note: Can receive credit for $$\texttt{DMA\_DST\_DEC}$$ or $$\texttt{DMA\_SRC\_DEC}$$ only if decrementing logic was implemented correctly in prior parts

+ **1.5 pts** Partial credit for the $$\texttt{.cnt}$$ flags: Specifies the correct flags but used commas or logical OR ($$\texttt{||}$$)

+ **2 pts** Correctly specifies the $$\texttt{.cnt}$$ width:

$$\texttt{(row + 1)}$$

Note: Off-by-one errors are acceptable (e.g. $$\texttt{row}$$)

- **1 pts** Deduction: Correct width with minor var. name error

i.e. $$\texttt{img}$$, $$\texttt{r}$$, etc.

4.5 Color portion: SRC 0 / 5

✓ **+ 0 pts Graded**

+ **5 pts** Correctly specifies the $$\texttt{.src}$$:

$$\texttt{\&color}$$

4.6 Color portion: DST 0 / 5

✓ **+ 0 pts Graded**

+ **5 pts** Correctly specifies the $$\texttt{.dst}$$:

$$\texttt{videoBuffer + OFFSET(row, row + 1, GBA\_WIDTH)}$$

i.e. $$\texttt{videoBuffer + (row * GBA\_WIDTH) + (row + 1)}$$

Note: Could have also used decrementing logic

i.e. $$\texttt{videoBuffer + (row * GBA\_WIDTH) + width - 1}$$

+ **2.5 pts** Partial credit for the $$\texttt{.dst}$$: Calculates the correct offset but passes a $$\texttt{u16}$$ instead of $$\texttt{u16*}$$

i.e. $$\texttt{videoBuffer[OFFSET(row, row + 1, GBA\_WIDTH)]}$$

- **1.5 pts** Deduction: Correct with minor var. name error

i.e. $$\texttt{img}$$, $$\texttt{r}$$, etc.

4.7 Color portion: CNT 0 / 5

✓ **+ 0 pts Graded**

+ **3 pts** Correctly specifies the $$\texttt{.cnt}$$ flags:

$$\texttt{\: \: - DMA\_ON}$$
$$\texttt{\: \: - DMA\_DST\_INC}$$
$$\texttt{\: \: - DMA\_SRC\_FIX}$$

Note: Can receive credit for $$\texttt{DMA\_DST\_DEC}$$ only if decrementing logic was implemented correctly in prior parts

+ **1.5 pts** Partial credit for the $$\texttt{.cnt}$$ flags: Specifies the correct flags but used commas or logical OR ($$\texttt{||}$$)

+ **2 pts** Correctly specifies the $$\texttt{.cnt}$$ width:

$$\texttt{(width - (row + 1))}$$
$$\texttt{(width - \: row - 1))}$$

Note: Off-by-one errors are acceptable (e.g. $$\texttt{(width - row)}$$)

- **1 pts** Deduction: Correct width with minor var. name error

i.e. $$\texttt{img}$$, $$\texttt{r}$$, etc.

CS 2110 — Quiz 6 (30 minutes)
November 14th, 2018

Full name: Izaan Kamal

GT username: ikamal3

This quiz is worth a total of **100 points**.

In accordance with the Georgia Institute of Technology Honor Code, I have neither given nor received aid on this quiz.

Signature: Izaan Kamal

Please make sure all of your answers are contained within the answer boxes or the fill-in lines.
You have been provided with scratch paper for your work. You will **NOT** be given credit for showing work. Having anything except the answer inside the boxes or above the fill-in lines might cause incorrect results.
Write your name and answers legibly. You will not receive credit for illegible answers.

Warning: All code you write MUST compile with the standard homework flags:

```
-std=c99 -pedantic -Wall -Werror -Wextra
```

What's the Point?

1. Consider the following code segment:

```
char       a[5][10];
short      b[25];
int        c[20];
struct s   d[10][20][30];
```

Using pointer arithmetic complete the following expressions. **You may not use [or]!**

(a) Extract the seventh `short` in `b`: [5]

`short shortValue =` (b + 24) ;

(b) Find the address of the second `int` in `c`: [5]

`int *intAddr =` &(c+4) ;

(c) Extract the `char` at `a[3][2]`: [7]

`char charValue =` (a+32) ;

(d) Find the address of the `struct s` at `d[5][7][10]`: [7]

```
struct s *pd = &d[0][0][0];
struct s *structAddr = pd +  3031 ;
```

Searching for a Book

2. The function `findBestBook` has three parameters: `books` (an array of Books), `size` (the number of Books in books) and `bookComp` (a user-supplied function for comparing two Books). [16]

Complete the function definition by filling in the correct parameter type for `bookComp`.

```
1
2  Book *findBestBook(Book *books, int size, Book* book(comp())
3  {
4    if ((!books) || (!bookComp)) return NULL;
5
6    Book *bestBook = &books[0];
7
8    for (int i = 1; i < size; i++)
9      bestBook = (*bookComp)(bestBook, &books[i]);
10
11   return bestBook;
12 }
```

Extracting Channels of a Pixel

3. Write a function `extractChannels` which takes a `u32` `pixel` (see diagram below) and returns the three color channels through `u32*` parameters `red`, `green`, and `blue`. Each color channel consists of 10 bits and the uppermost bits, `[31:30]`, are unused. *Note:* `u32` is an alias for `unsigned int` on ARM. [20]

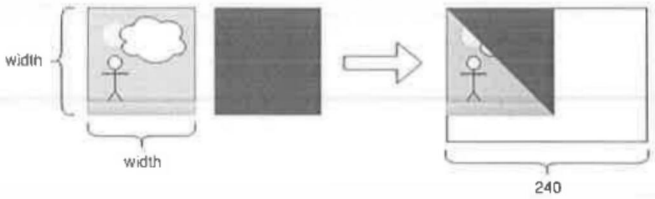
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		G	G	G	G	G	G	G	G	G	G	R	R	R	R	R	R	R	R	R	R	B	B	B	B	B	B	B	B	B	B

Reminder: The color channel parameters are pointers to 32-bit values!

```
1  void extractChannels(u32 pixel, u32 *red, u32 *green, u32 *blue)
2  {
3    blue= pixel & 0x3FF;
4
5    red   =pixel & 0x FFC00;
6    green =pixel & 0x 3FF00000;
7  }
```

Drawing a Collage with DMA

4. The function `drawSquareDiagonalCollage` collages a square `image` and a `color` along the image's diagonal: Row zero consists of one pixel of the `image` and the remainder of the `color`. The final row consists entirely of the `image`. [40]



Do **not** copy the full `image` or a full square of the `color`, only the portions appearing in the collage.

Note: You're allowed to call DMA for small copies.

```
#define GBA_HEIGHT 160
#define GBA_WIDTH  240

#define OFFSET(r, c, w) (((r) * (w)) + (c))

#define DMA_DST_INC  (0 << 21)
#define DMA_DST_DEC  (1 << 21)
#define DMA_DST_FIX  (2 << 21)
#define DMA_DST_RST  (3 << 21)

#define DMA_SRC_INC  (0 << 23)
#define DMA_SRC_DEC  (1 << 23)
#define DMA_SRC_FIX  (2 << 23)

#define DMA_ON       (1 << 31)
```

```
1  volatile unsigned short *videoBuffer = (unsigned short *) 0x6000000;
2
3  void drawSquareDiagonalMontage(const u16 *image, int width, u16 color)
4  {
5    for (int row = 0; row < width ; row++)
6    {
7      DMA[3].src = image ; // Draw the image portion
8
9      DMA[3].dst = ____________________________________________________________;
10
11     DMA[3].cnt = OFFSET(DMA_SRC_INC, DMA_DST_INC, DMA_ON)
12
13                  ____________________________; // Continue DMA[3].cnt here
14
15     DMA[3].src = color ; // Draw the rectangle portion
16
17     DMA[3].dst = ____________________________________________________________;
18
19     DMA[3].cnt = ____________________________________________________________
20
21                  ____________________________; // Continue DMA[3].cnt here
22   }
23 }
```