COMP1521 26T1

Week 2 Lecture 2

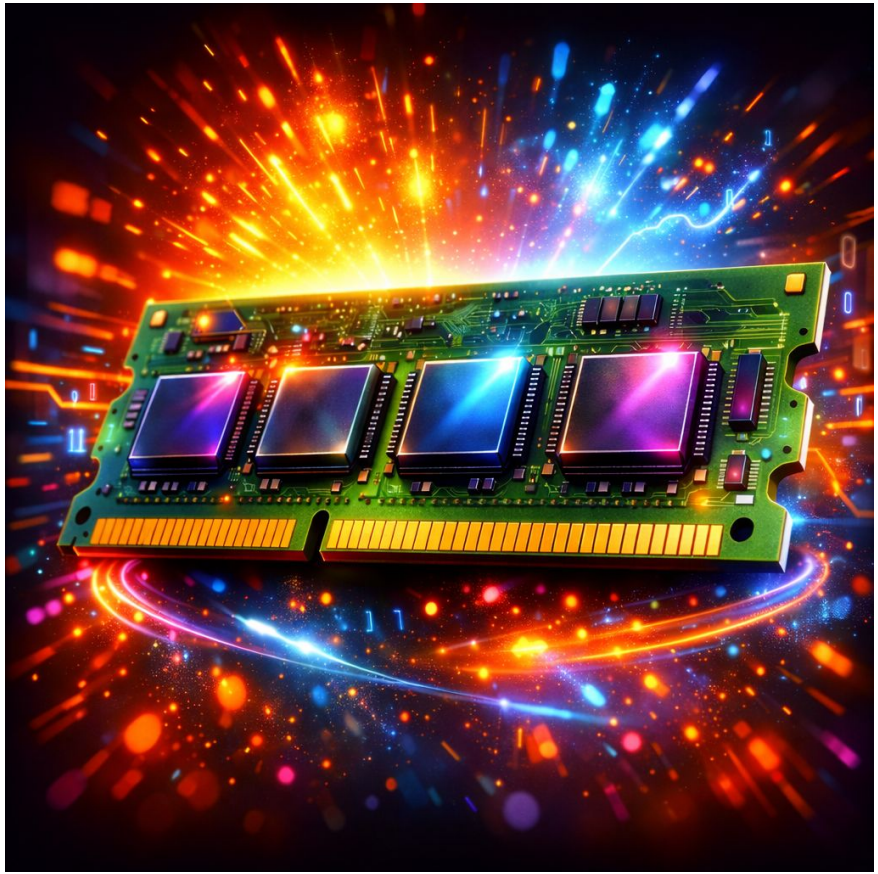
MIPS Data and Memory

Help Sessions and Revision Sessions

- Help Sessions:
  - Start next week
  - Schedule will be released this week.
  - Lab 1 and 2 due monday midday
- Revision Sessions:
  - Week 2 revision session was 6-8pm Today!!

Today's Lecture

- Recap Mondays lecture
  - Data and Memory
    - Global variables
    - Pointers
- 1D arrays
- 2D arrays
- C structs



Accessing Memory

- Loading data:
  - To perform computations, data must be transferred from memory into the CPU registers
  - `lb, lh, lw`

- Storing data:
  - Modified data must be written back from the CPU registers to memory
  - `sb, sh, sw`

- We can load and store whole bytes (not bits)

Recap: Global Variable Exercises

```c
int max = 45;

int main(void) {
    max--;
    printf("%d", max);
    putchar('\n');
    return 0;
}
```

```c
char x = 'Z';

int main(void) {
    x--;
    printf("%c", x);
    putchar('\n');
    return 0;
}
```

Pointer Example

```c
int answer = 42;

int main(void) {
    int i;
    int *p;
    p = &answer;
    i = *p;
    printf("%d\n", i);
    *p = 27;
    printf("%d\n", answer);
    return 0;
}
```

What would this print?

How could we write this in MIPS?

Mipsy assembler directives

```
.text                   # following instructions placed in text segment
.data                   # following objects placed in data segment


a: .space 18            # char a[18];
.align 2                # align next object on 4-byte addr
i: .word 42             # int32_t i = 42;
v: .word 1,3,5          # int32_t v[3] = {1,3,5};
h: .half 2,4,6          # int16_t h[3] = {2,4,6};
b: .byte 7:5            # int8_t b[5] = {7,7,7,7,7};
f: .float 3.14          # float f = 3.14; NOT USED IN THIS COURSE
s: .asciiz "abc"        # char s[4] {'a','b','c','\0'};
t: .ascii "abc"         # char t[3] {'a','b','c'};
```

Initialising a global array

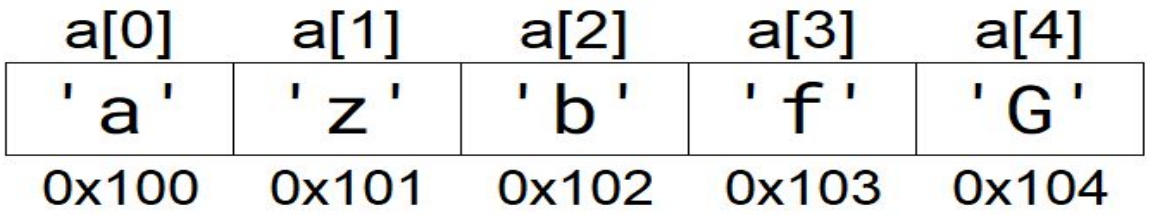
```
vec:   .space 40                    # int vec[10] or char vec[40]
nums:  .word 1, 3, 5, 7, 9          # int nums[] = {1,3,5,7,9}
nums2:  .word 1:4                   # int nums[] = {1,1,1,1}
str:   .byte 'a', 'b', 'c', '\0'    # char str[] = {'a','b','c','\0'}
str2:  .asciiz "abc"                # char str2[] = "abc"
chars:  .ascii "abc"                # char str2[] = {'a','b','c'}
```

How can we access elements?

How can we loop through the arrays?

Arrays of 1 byte elements

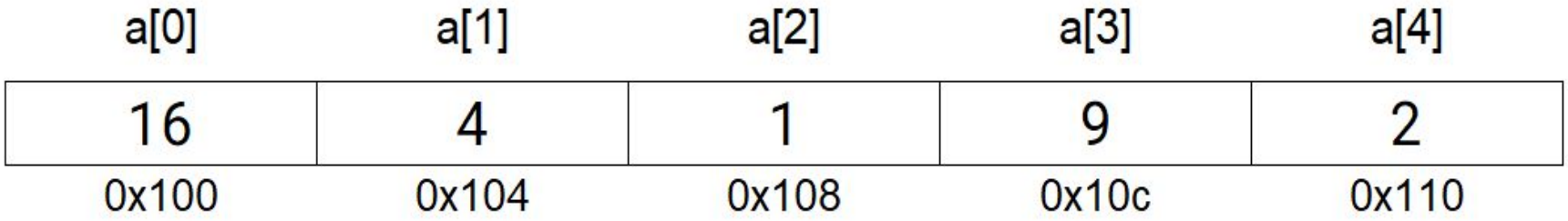
char a[5] = {‘a’, ‘z’, ‘b’, ‘f’, ‘G’};



- If we have the address of the **start of the array**:
  - How can I work out the address of the **a[3]**?
  - How can I work out the address of the **a[i]**?

Arrays of 4 byte elements

int a[5] = {16, 4, 1, 9, 2};



- If we have the address of the **start of the array**:
  - How can I work out the address of the a[3]?
  - How can I work out the address of the a[i]?

Address of Array Elements

char array: address of a[i] = address of a + i

integer array: address of a[i] = address of a + (i * 4)

In general:

address a[i] = address of a + i * sizeof(element)

Example: Accessing array elements in MIPS

```
        .data
my_char_array:
        .byte   'k', 'f', 'c'
my_int_array:
        .word   8, 99, 5
```

Let's translate C code to access the arrays at the given indexes and print them in MIPS

```
printf("%c", my_char_array[0]);
printf("%d", my_int_array[0]);
printf("%c", my_char_array[2]);
printf("%d", my_int_array[2]);
```

MIPS array coding examples

array_bytes_indexes.c

array_ints_indexes.c

Pointer Arithmetic in C (array.c demo)

In C adding 1 to a pointer increases it by the **sizeof** the type it points to!

This makes it easy to use a pointer to iterate through an array!

```
char   *p = 0x6060;  p++;  // (p == 0x6061)
int    *q = 0x6060;  q++;  // (q == 0x6064)
double *r = 0x6060;  r++;  // (r == 0x6068)
```

In MIPS we have to make sure we take this into account ourselves!

Pointer Arithmetic in C Warning

[Style Guide]

Rule | Explanation

Pointer Arithmetic

Avoid pointer Arithmetic.

Use array indices instead.

Experienced programmers use pointer arithmetic to produce succinct idiomatic code.

Novice programmers confuse themselves by trying to use pointer arithmetic. Any code using pointer arithmetic can also be written using array indices. Use array indices unless you are confident in your programming ability and are sure it is produces more readable code then array indices.

Pointer Arithmetic

```
int main(void) {
    fgets(array,ARRAY_LEN,stdin);
    char *p = array;
    while (*p != '\0') {
        printf("%c", *p);
        p++;
    }
    putchar('\n');
    return 0;
}
```

Pointer Arithmetic

```c
int main(void) {
    int *p = &array[0]; //same as array
    int *q = &array[4]; //same as array + ARRAY_LAST_INDEX
    while (p <= q) {
        printf("%d", *p);
        putchar(' ');
        p++;
    }
    putchar('\n');
    return 0;
}
```

2D Arrays in MIPS

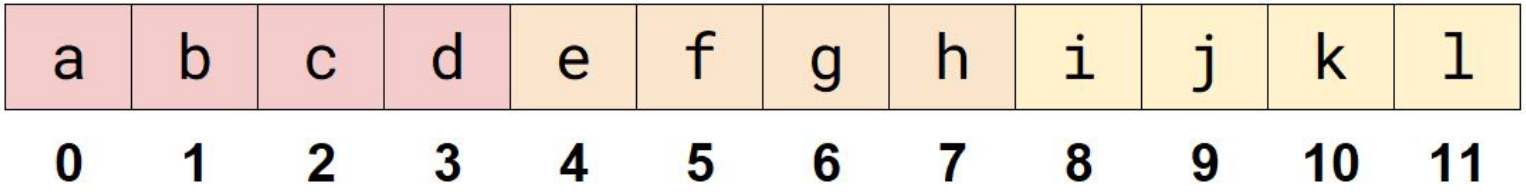
	0	1	2	3
0	a	b	c	d
1	e	f	g	h
2	i	j	k	l

< col

^ row

RAM is really just a 1D array.
A 2D array is really represented in memory with each row next to each other.

We need to map our 2 indexes to the appropriate offset



2D Arrays in MIPS

	0	1	2	3
0	a	b	c	d
1	e	f	g	h
2	i	j	k	l

< col

^ row

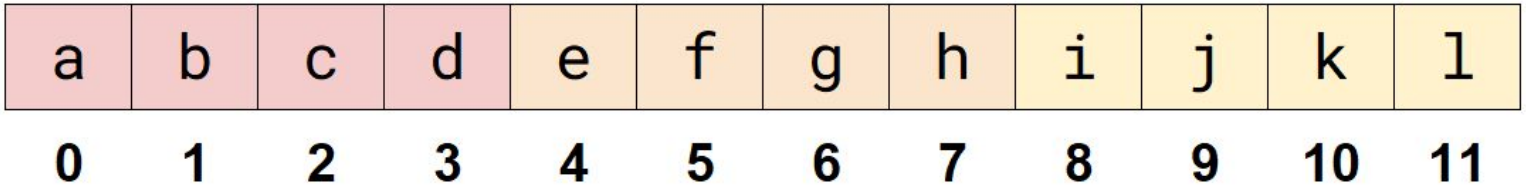
For an array of char

What would the offset be for:

a[1][0]

a[2][3]

What about for an array of int?



2D Arrays in MIPS

	0	1	2	3
0	a	b	c	d
1	e	f	g	h
2	i	j	k	l

< col

^ row

Offset of start of relevant row:
(row * N_COLS) * sizeof(element)

Offset within row:
col * sizeof(element)

Total offset:
(row * N_COLS + col) * sizeof(element)

a	b	c	d	e	f	g	h	i	j	k	l
0	1	2	3	4	5	6	7	8	9	10	11

MIPS 2d array coding examples

flag.c

print_2d_int.c

Structs in C: sizeof_struct.c

How size in bytes would these structs be?
Let's check our answers in C using sizeof

```c
struct s0 {
    int x;
    int y;
};
```

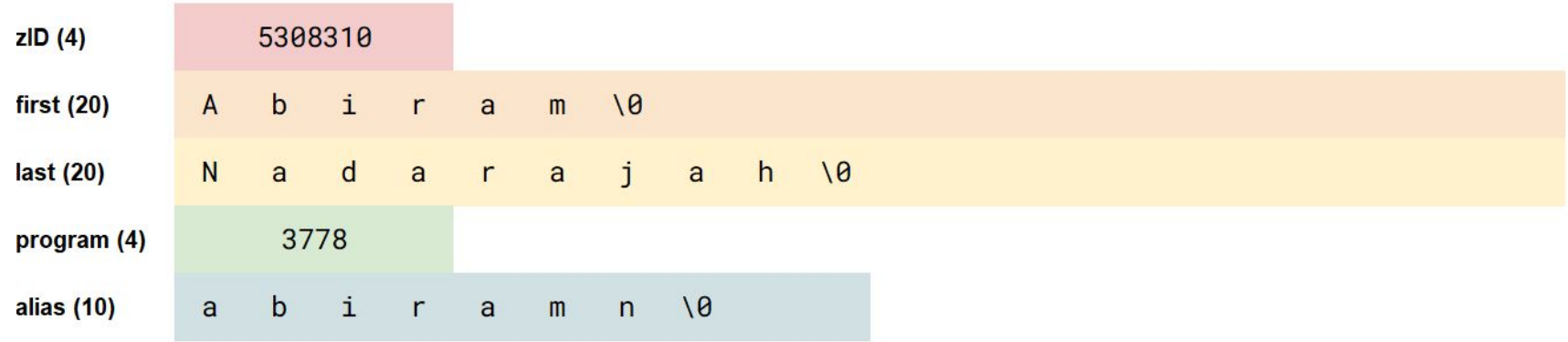
```c
struct s1{
    char c1;
    char c2;
};
```

```c
struct s2{
    char c1;
    char c2;
    int x;
};
```

```c
struct s3{
    int x;
    char s[3];
};
```

Structs: struct.c struct.s

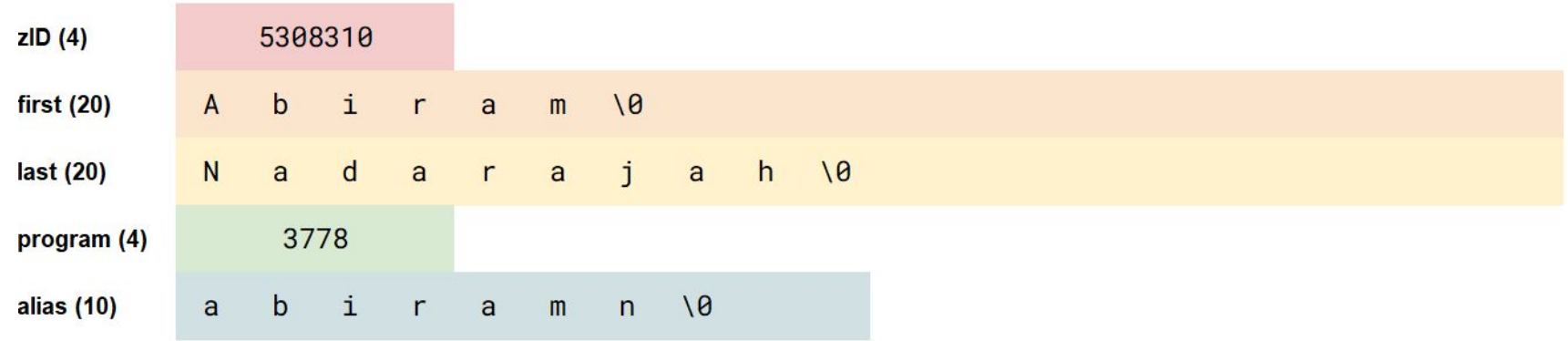
```
struct student {
    int zid;
    char first[20];
    char last[20];
    int program;
    char alias[10];
}
```



Structs: struct.c struct.s

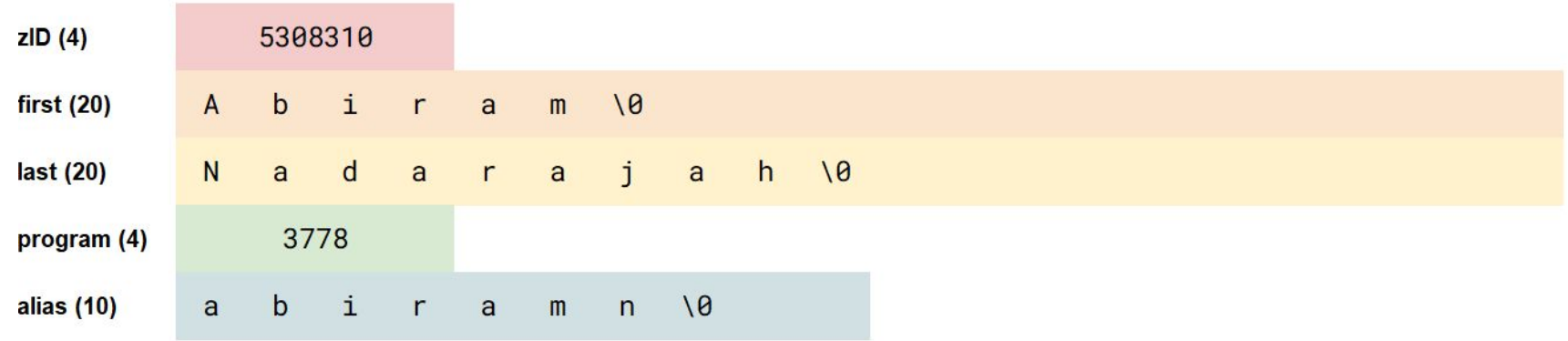
```
struct student {
    int zid;          //Offset 0
    char first[20];
    char last[20];
    int program;
    char alias[10];
};
```

structs are really just sets of variables at known offsets



Structs: struct.c struct.s

```
struct student {
    int zid;          //Offset 0
    char first[20];   //Offset 0 + 4 = 4
    char last[20];
    int program;
    char alias[10];
};
```

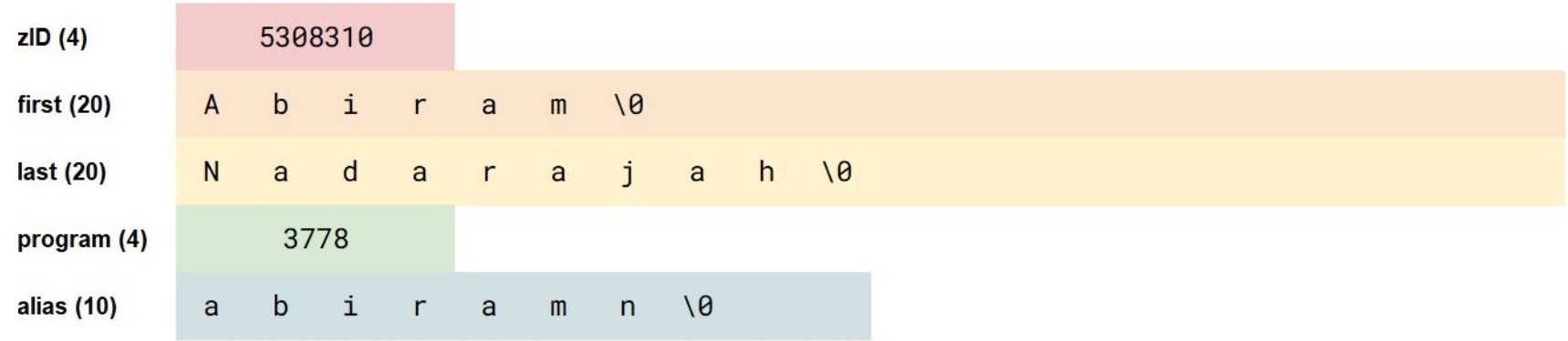


structs are really just sets of variables at known offsets

Structs: struct.c struct.s

```
struct student {
    int zid;          //Offset 0
    char first[20];   //Offset 4
    char last[20];    //Offset 4 + 20 = 24
    int program;
    char alias[10];
};
```

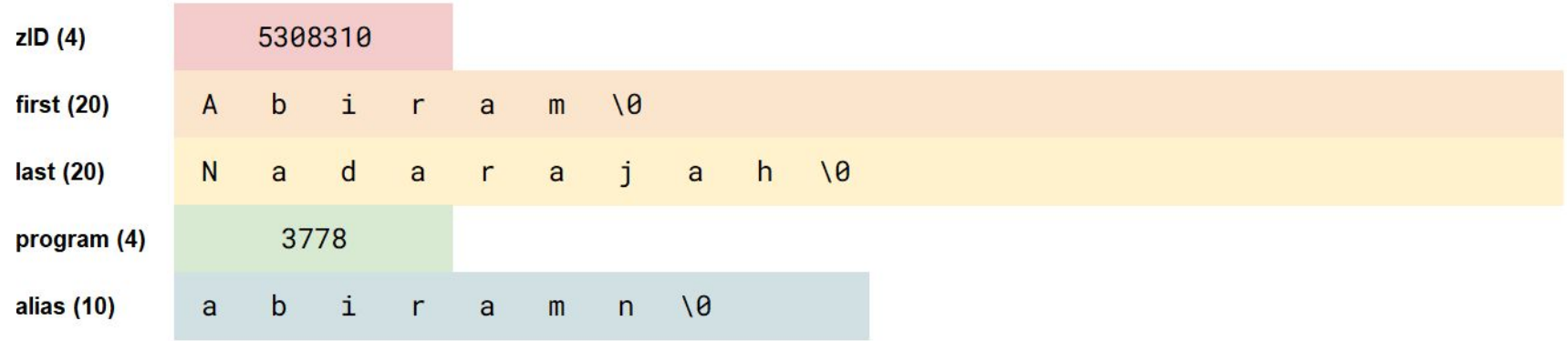
structs are really just sets of variables at known offsets



Structs: struct.c struct.s

```
struct student {
    int zid;          //Offset 0
    char first[20];   //Offset 4
    char last[20];    //Offset 24
    int program;      //Offset 24 + 20 = 44
    char alias[10];
};
```

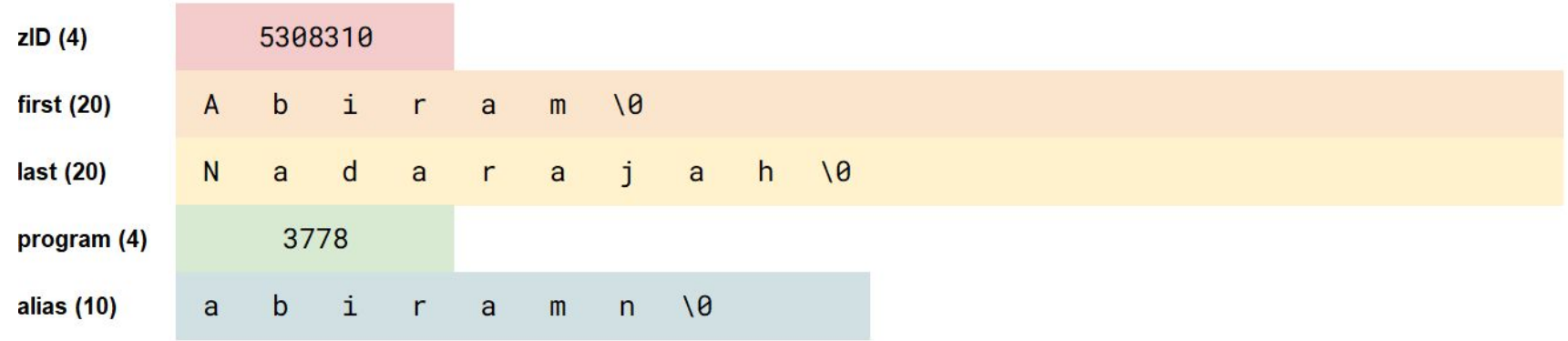
structs are really just sets of variables at known offsets



Structs: struct.c struct.s

```
struct student {
    int zid;          //Offset 0
    char first[20];   //Offset 4
    char last[20];    //Offset 24
    int program;      //Offset 44
    char alias[10];   //Offset 44 + 4 = 48
};
```

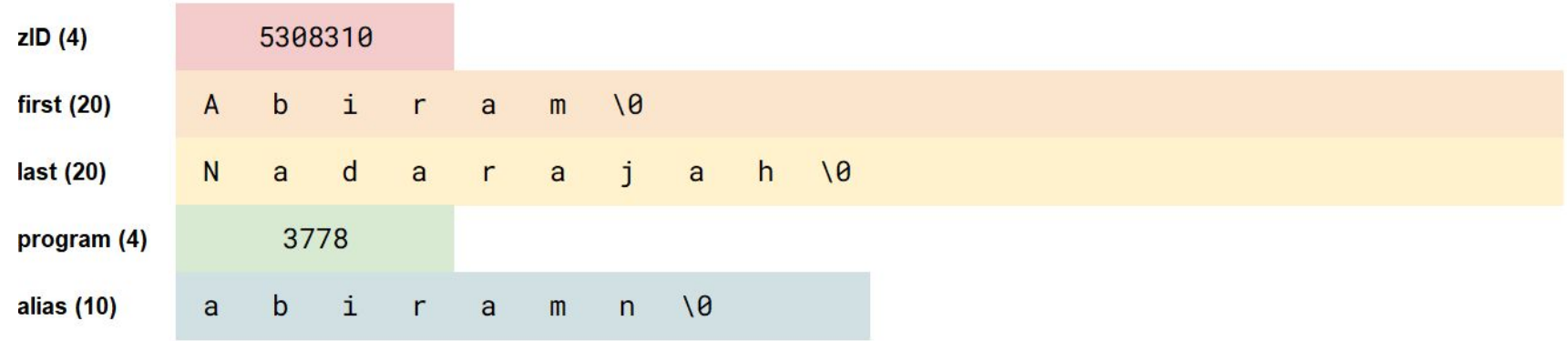
structs are really just sets of variables at known offsets



Structs: struct.c struct.s

```
struct student {
    int zid;          //Offset 0
    char first[20];   //Offset 4
    char last[20];    //Offset 24
    int program;      //Offset 44
    char alias[10];   //Offset 48
};// Total size: 48 + 10 + 2 (for alignment) = 60
```

structs are really just sets of variables at known offsets



What did we learn today?

- MIPS
  - recap of loading and storing data and pointers
  - arrays (1d and 2d)
  - structs
- Next lecture:
  - Functions in MIPS

Additional Resources

- MIPS Data Notes MIPS Data Code

Feedback Please!

Your feedback is valuable!

If you have any feedback from today's lecture, please follow the link below or use the QR Code.

Please remember to keep your feedback constructive, so I can action it and improve your learning experience.

https://forms.office.com/r/SF07w8WEAz

Reach Out

Content Related Questions:

Forum

Admin related Questions email:

cs1521@cse.unsw.edu.au

Student Support | I Need Help With…

My Feelings and Mental Health
Managing Low Mood, Unusual Feelings & Depression

Mental Health Connect
student.unsw.edu.au/**counselling**
Telehealth

Mind HUB
student.unsw.edu.au/**mind-hub**
Online Self-Help Resources

In Australia Call Afterhours UNSW Mental Health Support Line
1300 787 026
5pm-9am

Outside Australia Afterhours 24-hour Medibank Hotline
+61 (2) 8905 0307

Uni and Life Pressures
Stress, Financial, Visas, Accommodation & More

Student Support
Indigenous Student Support
student.unsw.edu.au/**advisors**

Reporting Sexual Assault/Harassment

Equity Diversity and Inclusion (EDI)
edi.unsw.edu.au/**sexual-misconduct**

Educational Adjustments
To Manage my Studies and Disability / Health Condition

Equitable Learning Service (ELS)
student.unsw.edu.au/**els**

Academic and Study Skills

Academic Language Skills
student.unsw.edu.au/**skills**

Special Consideration
Because Life Impacts our Studies and Exams

Special Consideration
student.unsw.edu.au/**special-consideration**