COMP1521 25T1

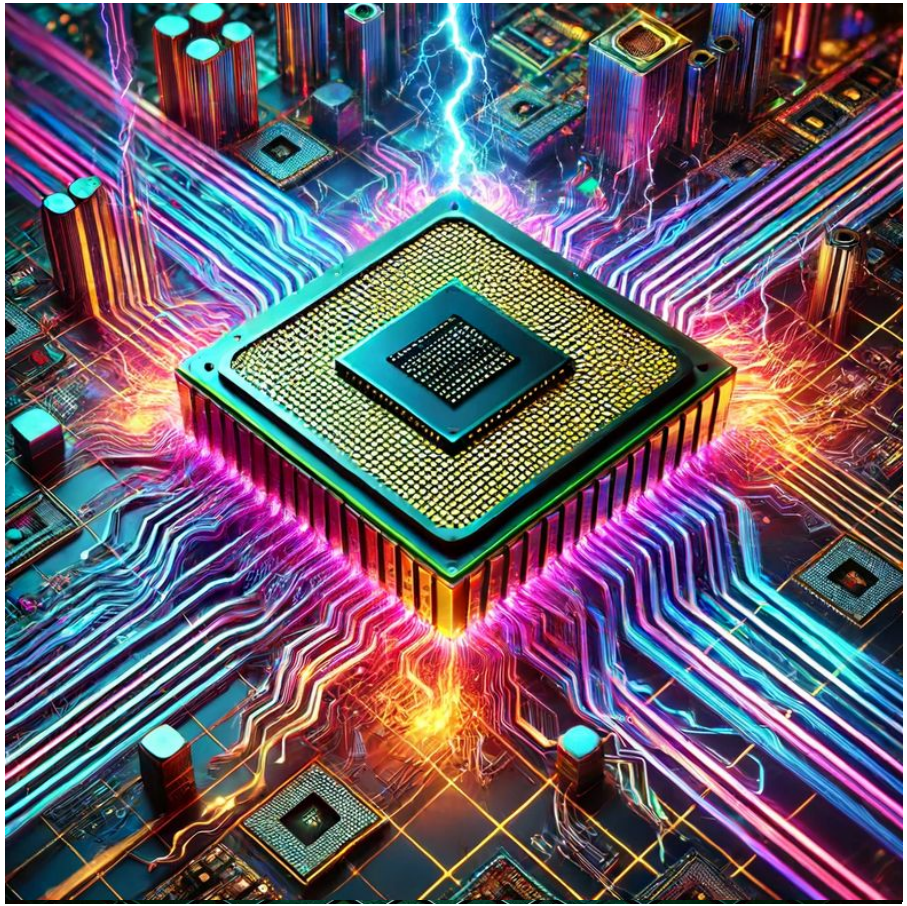
Week 3 Lecture 2

MIPS FUNctions and MIPS recap

Adapted from Abiram Nadarajah, Hammond Pearce,
Andrew Taylor and John Shepherd's slides

Today's Lecture

- More about Functions
- MIPS application
  - Putting it all together



Help Sessions

COMP1521 Help Sessions

Please check the schedule!
Optional, drop in help!
BYOD

They will get busy around assignment 1 deadline!
Get in now!

First Weekly Tests Out Tomorrow

Released: Thursday 3pm
Time limit: 1 hour
Due: Thursday Week 4 at 3pm. (And then another test comes out)
Submitted via **give**
You can get 50% max for questions submitted after the hour is up

Topic for week 3 test: MIPS basics, control.
Can use mips documentation

Functions

Functions - a summary

- Functions are named pieces of code (**labels**)
  - Which you can call
  - Which you can (optionally) supply arguments
  - Perform computations using those arguments
  - And return a value to a caller

Functions - a summary

- Functions are named pieces of code (**labels**)
  - Which you can call (`jal`)
  - Which you can (optionally) supply arguments
  - Perform computations using those arguments
  - And return a value to a caller

Functions - a summary

- Functions are named pieces of code (**labels**)
  - Which you can call (**`jal`**)
  - Which you can (optionally) supply arguments (**`$a0 - $a3`**)
  - Perform computations using those arguments
  - And return a value to a caller

Functions - a summary

- Functions are named pieces of code (**labels**)
  - Which you can call (**`jal`**)
  - Which you can (optionally) supply arguments (**`$a0 - $a3`**)
  - Perform computations using those arguments
  - And return a value (**`$v0`**) to a caller

Register Usage

Can use the other 30 registers however we want, technically, but:
There are conventions to prevent utter chaos and madness

Number	Names	Conventional Usage
0	zero	Constant 0
1	at	Reserved for assembler
2,3	v0,v1	Expression evaluation and results of a function
4..7	a0..a3	Arguments 1-4
8..16	t0..t7	Temporary (not preserved across function calls)
16..23	s0..s7	Saved temporary (preserved across function calls)
24,25	t8,t9	Temporary (not preserved across function calls)
26,27	k0,k1	Reserved for Kernel use
28	gp	Global Pointer
29	sp	Stack Pointer
30	fp	Frame Pointer
31	ra	Return Address (used by function call instructions)

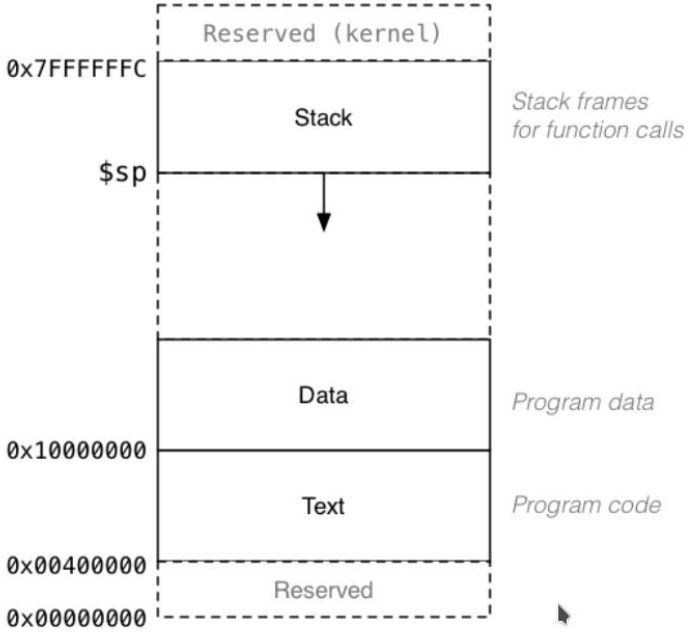
MIPS function rules: Summary

- **`$t`** registers are free real estate
  - So we must assume that other functions destroy them
- A function must restore the original values of **`$sp`**, **`$fp`**, **`$s0`**..**`$s7`**
  - So we can assume that any function we call leaves these registers unchanged
- Functions need to preserve **`$ra`** if they overwrite it
  - Otherwise, our function will lose track of where to return to
- **`$a0`**..**`$a3`** contain arguments -
  - these are also not preserved by callees (like $t)
- **`$v0`** contains the return value

Saving to the Stack

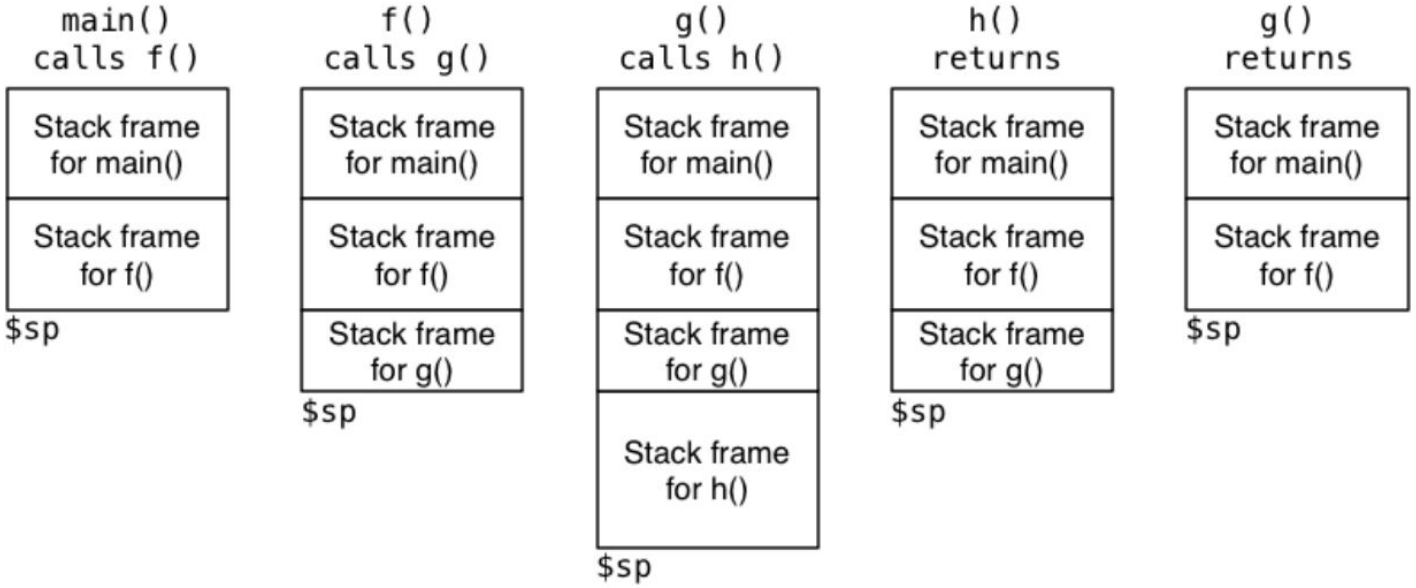
The stack

- is a region of memory which we can grow and expand
- uses the $sp (**stack pointer**) register to keep track of the top of the stack
- We can modify the stack pointer to allocate more room on the stack for us to store values



The stack: growing and shrinking

This is how the stack changes as functions are called and return:



Example - $sp and the stack (the easy way)

- For convenience, we provide you with two pseudo-instructions to interact with the stack: **push** and **pop**
- **push R_t**
  - ‘allocates’ 4 bytes on the stack ($sp = $sp - 4)
  - stores the value of R_t to the stack
- **pop R_t**
  - restores the value on the top of the stack into R_t
  - ‘deallocates’ 4 bytes on the stack ($sp = $sp + 4)

Prologues and Epilogues

Prologues: the start of a function's story

- We use the **begin** instruction (more on this soon)
- We need to **push** $ra onto the stack
- We **push** the values of any $s registers we want to use

Epilogues: the end of a function's story

- We restore (**pop)** any $s registers we saved to the stack, in reverse order
- We **pop** $ra
- We use the **end** instruction (more on this soon)
- We then return to the caller with `jr $ra`

Function Skeleton

```
func:
        # [header comment]
func__prologue:
        begin
        push    $ra
        push    $s0
        push    $s1

func__body:
        # do stuff

        li      $a0, 42
        jal     foo         # foo(42)

        # foo return val in $v0

# at the end of the function
func__epilogue:
        pop     $s1
        pop     $s0
        pop     $ra
        end

        jr      $ra
```

Recap Exercises

function_example_broken.s

sum_to.c

sum_to_r.c

MIPS Pizzeria Application

MIPS Pizzeria: Data Types

```c
// Written by Hammond Pearce
include <stdio.h>

struct pizza_t {
    char size[10];
    int price_cents;
};

struct pizza_t pizza_options[3] = {
    {"small", 300},
    {"medium", 550},
    {"large", 800}
};
```

MIPS Pizzeria: Main

```c
int main(void) {
    printf("The available pizza options are:\n");
    for (int i = 0; i < 3; i++) {
        increase_price(&pizza_options[i], 100);
        print_pizza_t(&pizza_options[i]);
    }
    return 0;
}
```

MIPS Pizzeria: Functions

```
void print_pizza_t(struct pizza_t *pizza) {
    printf("Size: %s, ", pizza->size);
    printf("price: %d cents\n", pizza->price_cents);
}

void increase_price(struct pizza_t *pizza, int increase_cents) {
    pizza->price_cents += increase_cents;
}
```

Don’t forget before jumping into MIPS

- For each function
  - Simplify function in C
  - Compile and rerun the program to check it still works
- Don’t change everything at once without testing!

Writing Code in MIPS

- Plan register usage
- Style - consistent naming of labels
- Indentation
- Comments - equivalent line of C Code for MIPS code.

Revision

If time:

array_words_pointer.c

array_bytes_pointer.c

modify_2d.c

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/iqq1fDbMK5

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**