Final Exam information:

Tuesday, 7pm-10pm, in graphic arts building

8 actual pages

2 pages MC (15 questions, worth one point each, 2-4 responses each), should be obvious, no all of the above and all of the above
- If you write explanation of why you chose something

6 pages free response

1 page of handwritten notes, digital is ok

Breakdown:
25% Midterm 1
1. Expected to read machine code (not asked to write machine code)
40% Midterm 2
1. Know the layout of a runtime stack
2. Not asked to write a complete function in assembly, just snipped
3. In subroutines, and traps and interrupts
4. Most points in functions, being able to write a program and locate it on a runtime stack
30% New stuff
5% random bits and whatnot

Machine Code - 10 points
Subroutines - 10 points
Traps and Interrupts - 9 points
Functions - 22 points
Pointers - 12 points
Dynamic Allocation - 8 points
Structures - 14 points

Total 100 points

problems to expect:

Symbol table (m2)

vector table (m1)

pointer problem (review problem)

Office Hours:
9am-1pm Monday and Tuesday

Midterm review:

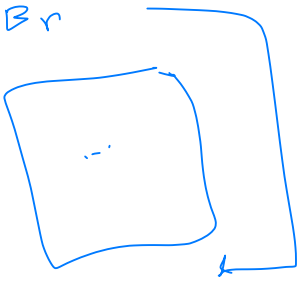
1. A programmer describes how to solve a problem in a high-level lang. Such as C

2. A compiler translates the high-level C into low-level assembly.
3. An assembler encodes the assembly in machine code
4. The hardware inherently understands machine code.

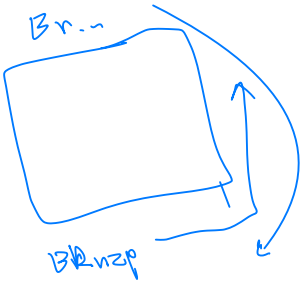
1. this is provably impossible for a computer (halting problem)

2. A compiler translates the high-level C into low-level assembly

Ex. if (...) { ... }



Ex. while (...) { ... }



In C, code is grouped into functions; each function application has local data.

Text segment — instructions

Data segment

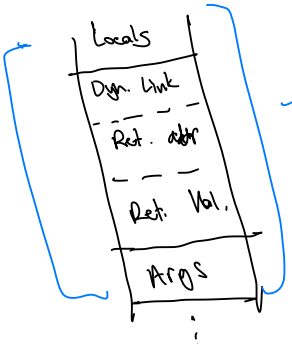
bss — block specialized

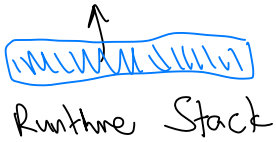
Heap

Each function application's data is in a stack frame.

Registers are temp. storage

- Locals are on the stack
  - → Globals (data segment)
- Dynamically allocated data is on the heap (programmers responsibility)





Runtime Stack

↳ Every location in memory has an address

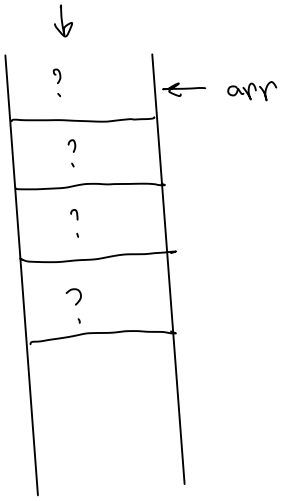
→ a pointer contains the address of another variable

(just a number where something is

In C, arguments are pass-by-value, but the value of a pointer is a reference

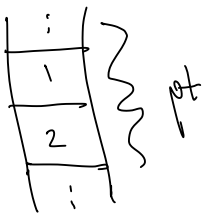
→ arrays are contiguous blocks; the value of an array is its first address

Ex `int arr[4];`



Ex
```
struct point {
    int x, y;
};

struct point pt = {1,2};
```



- Structures are contiguous blocks; the value of a struct is the whole struct

An assembler encodes the assembly in machine code. In a modern, stored program computer, instructions are data, encoded in binary for ease of implementation.
An ISA documents the binary machine instructions:
- Each instruction has a unique opcode
- Each instruction has ≥ 0 Operands

n bits can encode $\leq 2^n$ values

Ex: 4-bit opcodes $\therefore 2^4 = 16$ instructions

Ex: 8 registers $\therefore \log_2 8 = 3$ bits to identify register

* Changing the ISA requires changing the physical hardware

The hardware inherently understands machine code.

→ Fetch, decode, execute ... over and over again