Lecture 3

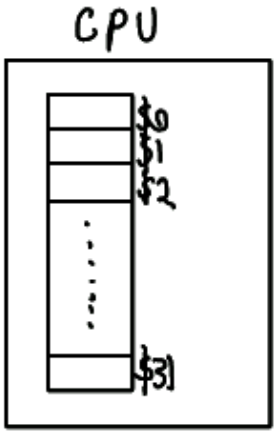
Assembly language

Say something once, why say it again?

CS 241: Foundations of Sequential Programs
Fall 2014

Troy Vasiga et al
University of Waterloo

Review



- MIPS has 18 instructions
- CPU contains 32 32-bit registers
- MIPS reference sheet for encoding of instructions
- `cs241.wordasm` to create binary files
- `mips.twoints` to start MIPS machine
  - reads $1, $2

Special Registers

- $0] — always 0
- $29]
- $30] — special
- $31] — initially contains your return address,
  where to go to (PC value)
  when I am finished executing

Assembly language

- encoding details can be automated
- 1-1 correspondence with machine language (almost)
- easier on the human eye
- uses different version of the assembler

Example 0 revisited

- A1 level (machine language):

```
00000000 10100111 00011000 00100000
00000011 11100000 00000000 00001000
```

⇒ hex {.word 0x.....

- A2 level (assembly language):

```
add $3, $5, $7
jr $31
```

— programming

- Which one would you rather read and debug?

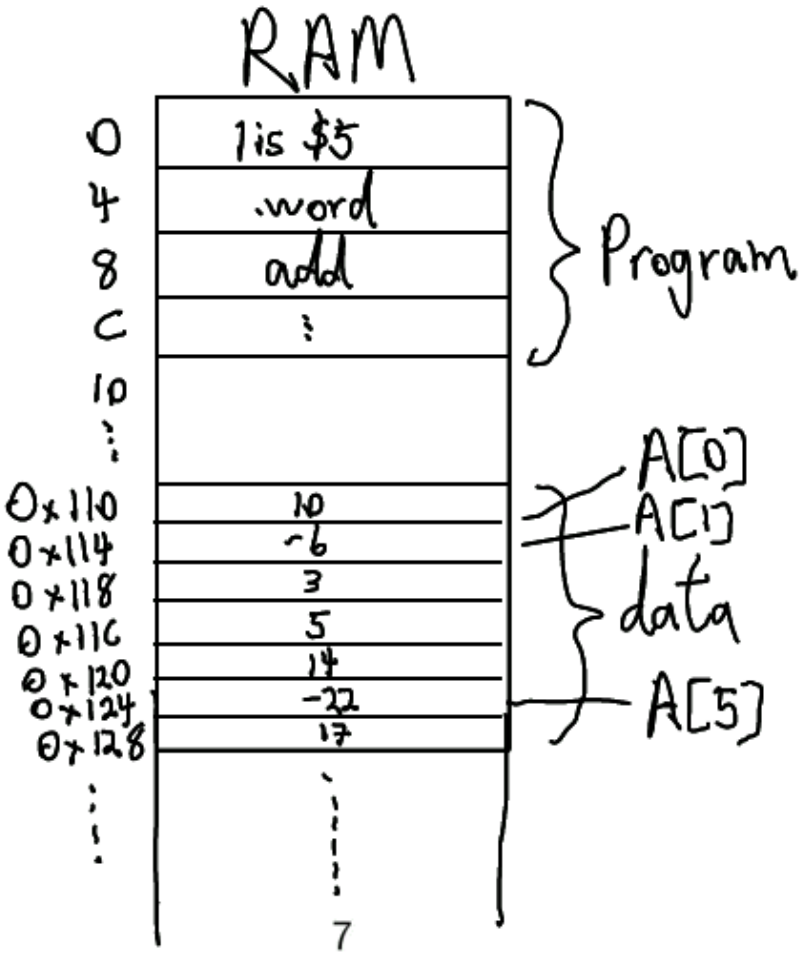
A3+A4 ⇒ write an assembler

Examples 1,2,4

- Example 1 – adding two immediate values
- Example 2 – finding absolute values] if statement (conditional evaluation)
- Example 4 – loop to sum integers from 1 to 13 → iterate / repeat

Turing Completeness

Memory Revisited



Example 3

- array access

Example 5 – Output

There is no Input