CS240

January 7, 2013

Assembly Language

```
1    lw $t0, 0($s0)
2    add $t0, $t0, $t0
3    sw $t0, 0($s0)
```

High Level Languages

```
1 int income = 200;
2 income = income * 2;
```

High Level Languages

```
1 int income = 200;
2 income = income * 2;
```

- data type: A collection of values along with a collection of operations for manipulating those values (textbook).
- data type: An interpretation of a sequence of bits, along with a set of operations that conform to that interpretation (mine).
- Data types may be either
  - primitive - built into the language
  - programmer defined

Abstract Data Type

- Abstract Data Type: a programmer-defined data type that specifies a set of data values and a collection of well-defined operations that can be performed on those values. Abstract data types are defined independently of their implementation.
  - simple: one or a few named fields
  - complex: a collection of data values

ADT Example: Date

- Date(month, day, year)
- day()
- month()
- year()
- monthName()
- dayOfWeek()
- numDays(otherDate)
- isLeapYear()
- ...

ADT Example: Bag

- Bag()
- length()
- contains(item)
- add(item)
- remove(item)
- iterator()

Data Structures

Data Structure: The actual data organization that underlies the implementation of a (complex) abstract data type.

Python Pros

- Minimal, easy to read syntax
- Comprehensive standard libraries
- Widely used
- Free and open source
- Dynamically typed
- Interpreted
- Stylistically flexible - OO, procedural, functional
- Fun

Python Cons

- Many of the data structures we will study are included as primitive types:
  - Python lists and dictionaries
- Undergoing a transition from 2.x $\rightarrow$ 3.x
- Dynamically typed
- Stylistically flexible
- Not particularly fast