

CS 241 – Week 5 Tutorial

Writing an Assembler, Part 2 & Regular Languages Part 1

Fall 2018

1 Assembling instructions automatically

Recall that instructions follow two basic formats, register and immediate, as detailed in the MIPS reference sheet:

```
Register: 0000 00ss ssst tttt dddd d000 00ff ffff
Immediate: oooo ooss ssst tttt iiii iiii iiii iiii
```

In both cases, `sssss`, `ttttt`, `dddd` are registers encoded as 5-bit unsigned numbers, `iiii iiii iiii iiii` is a two's complement 16-bit number, and `oooooo` or `ffffff` are 6-bit opcodes specified for each instruction in their row on the sheet.

Using the bitwise operations from last week's tutorial and the ideas from assignment 1, give pseudocode to assemble the following instructions:

1. `slt $d, $s, $t`.
2. `beq $s, $t, $i`, where i is an immediate value (INT or HEXINT token).

How could we design our code to maximize code reuse between various instructions?

2 Regular Languages Review

An alphabet (denoted Σ) is a finite set of symbols.

- $\{a, b, c\}$
- $\{b\}$
- $\{to, be, or, not, _ \}$
- $\{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$

A word (also called a string) is finite sequence of symbols from Σ , such as:

- bac, aba, c given that $\Sigma = \{a, b, c\}$
- ϵ, b, bb, bbb given that $\Sigma = \{b\}$
- $to_be_or_not_to_be$ (one word formed from the alphabet)
 $\Sigma = \{to, be, or, not, _ \}$
- $DEADBEEF, FACE, ABAC1234$ given that $\Sigma = \{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$

A language is a set of words.

A Regular Language R is a sets of words where either:

- R is the empty language
- R contains a single word
- R is the union of two regular languages
- R is the concatenation of two regular languages
- $R = L^* = \cup_{i=0}^{\infty} L^i$ where L is a regular language, $L^0 = \{\varepsilon\}$ and for $i > 0, L^i = L \cdot L^{i-1}$

Regular Expressions

Regular expressions are a means of expressing regular languages using combinations of symbols and specialized operations:

- Concatenation (ab) - a matching word has a followed by b
- Alternation ($a|b$) - a matching word has a or b but not both
- Repetition (a^*) - a matching word has 0 or more occurrences of a

Furthermore, we can group expressions into subexpressions using parenthesis. For example, $a(a|b)^*$ matches an a followed by 0 or more a 's and b 's. Note that this is all essentially just shorthand for the rather verbose set notation for regular languages.

3 Regular Expression Problems

Build the following languages using combinations of finite languages with regular operations (set notation):

1. Construct the language of binary strings whose 2nd letter is a '0' and whose 5th is a '1'.
2. Construct the language of binary strings that contain the substring "110101".

Provide a regular expression for each of the following languages:

1. Convert your solution to the 2nd regular language problem above into a regular expression.
2. $\Sigma = \{a, b\}, L = \{aa, ab, ba, bb\}$
3. Strings over the alphabet $\Sigma = \{a, b, +, -, \cdot, /\}$ representing valid arithmetic expressions with no parentheses. All operators should be binary (thus $a + -b$ is not valid) and multiplication must be written explicitly (thus $a \cdot b$ is valid but ab is not).
4. (*Difficult*) $\Sigma = \{0, 1, 2\}, L = \{x \in \Sigma^* | x \text{ contains an even number of 0's and at least one 1.}\}$

4 Deterministic Finite Automata (DFAs)

A Deterministic Finite Automaton (DFA) is a 5-tuple $(\Sigma, Q, q_0, \mathcal{A}, \delta)$ where:

- Σ – a non-empty finite set of symbols, the input alphabet
- Q – a non-empty finite set of states
- $q_0 \in Q$ – the starting state
- $\mathcal{A} \subseteq Q$ – a set of accepting states
- $\delta : Q \times \Sigma \rightarrow Q$ – the transition function

In the definition, $\delta(q, \sigma)$ exists for every $q \in Q$ and $\sigma \in \Sigma$. However, when drawing the DFA diagram, we often assume that the transition from one state goes to error state if it was not shown in the diagram. An error state is any state that can never reach an accepting state by consuming any inputs, and error state itself can not be an accepting state.

4.1 DFA Problems

Draw DFA diagrams for the following languages:

1. The language of strings over $\Sigma = \{a, b, c\}$ that contain exactly one a and an even number of c 's (no restriction on number of b 's).
2. The language of strings over $\Sigma = \{a, b\}$ that contain an even number of a 's and an odd number of b 's. How would the solution change if we added c to the alphabet and words could have any number of c 's?
3. The language of strings over $\Sigma = \{0, 1\}$ that end in 1011. How would the solution change if 1011 could appear anywhere in the string?
4. The language of strings over $\Sigma = \{0, 1, 2, 3\}$ which are integers whose digit sum is 3. Leading zeros are permitted.
5. The language of strings over $\Sigma = \{a, b, c\}$ that end in cab and contain an even number of a 's (no restriction on the number of b 's or c 's).