CS1101: Lecture 39

The Assembly Process – Pass Two

Dr. Barry O'Sullivan
b.osullivan@cs.ucc.ie

Course Homepage
http://www.cs.ucc.ie/~osullb/cs1101

Lecture Outline

- Pass Two
- Code Generation
- Errors
- Managing the Symbol Table
- Arrays
- Binary Search
- Hash Coding
- **Reading**: Tanenbaum, Chapter 7, Section 3.

Pass Two

- The function of **pass two** is to generate the object program and possibly the assembly listing.
- In addition, it must output certain information by the linker for linking up procedures assembled at different times into a executable file.
- Each line is read and processed one at a time
- Since we have written the type, opcode, and length at the start of each line (temporary file), these are read in to save some parsing.

Code Generation

- During **code generation** evaluation procedures are used to handle particular patterns in the assembly language.
- For example, an opcode and two register operands.
- Each one generates binary code for the relevant instruction.
- Normally, as code generation progresses, the binary code is buffered as it accumulates binary code and written to the disk in large chunks to reduce disk traffic.
- The original source statement and the object code generated from it (in decimal) can then be printed or put into a buffer for later printing.
- After the ILC has been adjusted, the next statement is fetched.

Errors

- Up until now it has been assumed that the source program does not contain any errors.
- Some of the common errors are as follows:

1. A symbol has been used but not defined.
2. A symbol has been defined more than once.
3. The name in the opcode field is not a legal opcode
4. An opcode is not supplied with enough operands.
5. An opcode is supplied with too many operands.
6. An octal number contains an 8 or a 9
7. Illegal register use (e.g. a branch to a register).
8. The END statement is missing.

Managing the Symbol Table

- During pass one of the assembly process, the assembler accumulates information about symbols and their values that must be stored in the **symbol table** for lookup during pass two.
- Several different ways are available for organizing the symbol table.
- All of them attempt to simulate an **associative memory**, which conceptually is a set of (symbol, value) pairs.
- Given the symbol, the associative memory must produce the value.

Arrays

- The simplest implementation technique is indeed to implement the symbol table as an array of pairs, the first element of which is (or points to) the symbol
- The second of which is (or points to) the value.
- Given a symbol to look up, the symbol table routine just searches the table linearly until it finds a match.
- This method is easy to program but is slow, because, on the average, half the table will have to be searched on each lookup.

Binary Search

- Another way to organize the symbol table is to sort it on the symbols and use the **binary search** algorithm to look up a symbol.
- This algorithm works by comparing the middle entry in the table to the symbol.
- If the symbol comes before the middle entry alphabetically, the symbol must be located in the first half of the table.
- If the symbol comes after the middle entry, it must be in the second half of the table.
- If the symbol is equal to the middle entry, the search terminates.

Binary Search

- Assuming that the middle entry is not equal to the symbol sought, we at least know which half of the table to look for it in.
- Binary search can now be applied to the correct half, which yields either a match, or the correct quarter of the table.
- Can search a table size n in $log_2 n$ attempts.

Hash Coding

- A completely different approach is known as **hash coding**.
- This approach requires having a "hash" function that maps symbols onto integers in the range 0 to $k-1$.
- One possible function is to multiply the ASCII codes of the characters in the symbols together, ignoring overflow, and taking the result modulo k or dividing it by a prime number.
- In fact, almost any function of the input that gives a uniform distribution of the hash values will do.

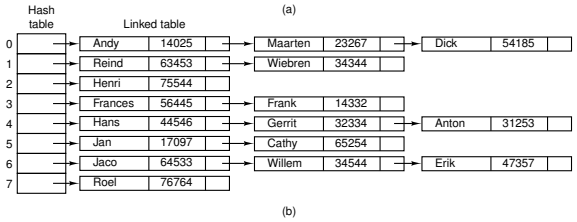
Hash Coding

- Symbols can be stored by having a table consisting of k **buckets** numbered 0 to $k-1$.
- All the (symbol, value) pairs whose symbol hashes to i are stored on a linked list pointed to by slot i in the hash table.
- With n symbols and k slots in the hash table, the average list will have length n/k.
- By choosing k approximately equal to n, symbols can be located with only about one lookup on the average.
- By adjusting k we can reduce table size at the expense of slower lookups.

Hash Coding

Andy	14025	0
Anton	31253	4
Cathy	65254	5
Dick	54185	0
Erik	47357	6
Frances	56445	3
Frank	14332	3
Gerrit	32334	4
Hans	44546	4
Henri	75544	2
Jan	17097	5
Jaco	64533	6
Maarten	23267	0
Reind	63453	1
Roel	76764	7
Willem	34544	6
Wiebren	34344	1

(a)



(b)