# INFORMATION THEORY METRIC FOR ASSEMBLY LANGUAGE

Curtis R. Cook
Computer Science Department
Oregon State University
Corvallis, Oregon 97331-3202

90-60-18

# INFORMATION THEORY METRIC FOR ASSEMBLY LANGUAGE

Curtis R. Cook
Computer Science Department
Oregon State University
Corvallis, Oregon 97331-3202

## ABSTRACT

This paper presents a study of the distribution of assembly language instructions in two avionics software applications and the development of an information theoretic software complexity metric based on the study. The instructions were grouped into classes, e.g load and store, arithmetic, jump, compare, etc. As expected the distribution of instruction classes over the modules was generally uniform and nearly three-fourths of the instructions were load, store, or jump instructions. The biggest surprise was that nearly one-half of the 320 different instructions were not used.

The assembly language complexity metric is based on the premise that programs with familiar instructions are much easier to understand and work with than programs with unfamiliar instructions. Familiarity was approximated by frequency of instruction use. The metric is similar to one developed by Berlinger, except we computed the information content per instruction class and the average information content per instruction while he summed the information content of the instructions. Maintenance programmers selected what they felt were the most difficult modules. Our metric gave the highest complexity values to most of these modules.

## INTRODUCTION

Software complexity metrics measure the ease or difficulty a programmer will experience in working with (e.g. testing, maintaining, understanding) a program. The most common complexity metrics are based on source code characteristics such as size (lines of code), control flow (number of different execution paths through program), data structures, information flow between modules, and combinations of characteristics [2].

A quite different class of software complexity metrics are those based on information theory. These measures consider the frequency of use of programming language elements. The basic premise of information theory metrics is that a programmer is more likely to accurately use or more quickly understand commonly used elements that infrequently used ones. For example, a typical assembly language has several hundred instructions including many with various options. These instructions can be partitioned into three subsets (frequent, less frequent, and rarely) according to how often they are used. The frequent subset is the set of instructions (e.g. load, store, shift, and compare) commonly used by assembly language programmers. Assembly language programmers thoroughly understand this subset. The rarely used subset of instructions are normally the infrequently used instructions such as the special purpose instructions (e. g. interrupt handling, channel, and chaining) or specialized forms of instructions. Because they are so rarely used these instructions are generally not well understood. The remainder of the instructions fall into the less frequently used subset.

The proportion of assembler instructions in the frequently, less frequently and rarely used subsets depends on the programmer and the number of different instructions. If there are a small number of different instructions normally they will be evenly split among the three subsets. If there are a large number of different instructions then a small proportion (e.g. 15-20%) will be used frequently while a large proportion (50% or more) will seldom be used.

It is important not to infer that the instruction mix in each of the three subsets will be the same for all programmers. Generally the mixture will be similar, but it may be quite dissimilar because of programmer preferences, the programming application domain, problem constraints, and so forth.

There is an interesting information theory analogy with written prose comprehension. Frequency of words is an important factor in measuring written English prose comprehension. Reading speed is slowed by unfamiliar words and readers will experience difficulty understanding prose with unfamiliar words. It takes longer for working memory to process unfamiliar words and determine the relationship between other units in the sentence. [5] Readability formulas assign an index of probable difficulty for readers and/or a grade level to prose. Researchers consider word difficulty the most important factor in readability. Hence readability formulas take into account word difficulty either by word

length (e.g. longer words are typically less frequently used), by a count of the number of words not in a list of common words or by assigning a weight to words according to their position in a list of common words [4]. Howes and Solomon [ 3] showed that frequently used English words are recognized faster than less frequently used words. However, general word lists fail to account for individuality as people with interests in particular fields use words in that field more frequently than they are found in general prose. Researchers believe that familiarity is a more important factor in measuring readability than frequency.

Information theory based software complexity metrics are more sensitive to the infrequently used program elements than frequently used ones. Programs with rarely used program elements are typically more difficult to understand and more likely to contain errors than programs with mostly frequently used program elements and no rarely used elements. Hence these measures would assign a higher complexity to an assembly language program with several rarely used instructions than to a program without rarely used instructions. This agrees with our intuition that a programmer encountering one of the rarely used instructions may need to refer to a reference manual for its details in order to understand its function.

In this paper we present an analysis of the distribution of assembly language instructions for two medium size aircraft avionics applications and an information theory software complexity metric based on the analysis. The software is decomposed functionally and each function is composed of files. The analysis showed a strong relation between distribution of instructions and the functions and is consistent over the two applications and their modified versions. Our software complexity metric weights instructions according to frequency of use. The files rated as most difficult by the programmers who maintain the avionics software were the files with high metric values.

## INSTRUCTION DISTRIBUTION FOR AV8B AND F18 SOFTWARE

The data available for this study was five aircraft avionics application programs written in CMS-2M with embedded AYK-14 Assembly language. Most of the program was in assembly language. The programs were designed as a set of functions each composed of files (modules). The file naming convention was that all files for a particular function began with the same first letter. For example, the mathematical files all begin with "M". This allowed the logical grouping of files by function.

We were also able to group the assembly language instructions into classes. The reference manual divides the AYK-14 Assembly Language instructions into 11 classes: (1) Load and Store, (2) Arithmetic, (3) Logical, ( 4) Compare, (5) Jump, (6) Shift, (7) Miscellaneous, (8) Executive Mode, (9) Command and Chain,

(10) Stack and Queue, and (11) Floating Point and Sort. This breakdown is typical for most assembly languages.

We developed a analysis program that tabulated the instruction frequencies for each file. Three of the five avionics programs in our data set were for the AV8B and two for the F-18. Version 7 of the AV8B was a later release of Version 5 and Version 6 was for night flying and developed from Version 5. For the F18, 89A is a later release of 87X.

The instruction frequencies and percentages for files and instruction classes for the AV8B and F18 programs are given in Tables 1 - 10. Tables 1, 3, 5, 7, and 9 give the instruction frequency for each instruction and function as well as the totals for each. The last row in these tables gives the percentage of the total instructions for each instruction class. For each function Tables 2, 4, 6, 8, and 10 give the percentage of instructions in each class for that function. Our analysis of the data is divided into two parts: (1) Instruction Distribution and (2) Outliers and Rarely Used Instructions.

## Instruction Distribution

1. There were no floating point (actually trigonometric and logarithmic functions) or sort instructions in any of the programs so they were omitted from the tables. Hence there were only 10 instruction classes.

2. As expected, the instruction distribution for both the AV8B and F18 were very similar. Over one-half of the instructions were load or store instructions and nearly one-fourth were jumps. Slightly over 10% were arithmetic and logical and 10-12% were compare and shift instructions. Miscellaneous was between 3 and 4% while executive, chain/command, and stack/queue totaled 1% or less. There were only 4 stack/queue instructions in the AV8B software and only 5 in the F18. The rank order of the instruction classes by frequency for the three AV8B data sets are identical and the rank order by frequency for the two F18 data sets are also identical. The rank orders for all five data sets are the same except for two transposes in the ordering: The arithmetic and compares; the shift and miscellaneous.

3. There are 320 distinct instruction op codes in the AYK-14 Assembly Language. In each of the five programs slightly more than one-half (169-175)) of the distinct op codes were used. Even though the load and store instruction class account for over half of the instructions used and there are 44 distinct load and store op codes, it was surprising that op code L (load register) and S (store register) accounted for over one-fourth of the program instructions.

4. The F18 files contained 50% more instructions than the AV8B files. There is quite a range of file size. The smallest files contain less than 100 assembly language instructions and the largest contain over 15,000 instructions.

**Outliers and Rarely Used Instructions**

1. Outliers: Outliers are defined for the percentage values in tables 2, 4, 6, 8, and 10. A value in one of these tables is an outlier if it is at least two standard deviations above the weighted average (last row in Tables 1, 3, 5, 7, and 9).

Load/Store: None.
Arithmetic: M and Q files (AV8B) and M files (F18).
Logic: Q files (F18).
Compare: Y files (AV8B).
Jump: None.
Shift: A files (AV8B).
Miscellaneous: B and Y files (AV8B) and B files (F18).

2. We operationally define a rarely used instruction class as one that accounts for at most 1% of the total number of instructions in the software. There are three rarely used instruction classes and these instructions are concentrated in a very few files:

Executive: Less than 200 occurrences. Used only in C, G, and X files for the AV8B data and used only in C, X, and Z files for F18 data.

Chain/Command: Used about 1%. For AV8B used only in C and X files. For F18 used only in C, D, X, and Z files.

Stack/Queue: Used 4 times in X files in AV8B and 5 times in X files in F18.

| Name | load | arith | logic | comp | jump | shift | misc | exec | chain | stack | total |
|---|---|---|---|---|---|---|---|---|---|---|---|
| afiles | 238 | 89 | 1 | 3 | 49 | 46 | 0 | 0 | 0 | 0 | 426 |
| bfiles | 633 | 100 | 85 | 147 | 280 | 78 | 170 | 0 | 0 | 0 | 1493 |
| cfiles | 470 | 63 | 51 | 80 | 272 | 30 | 17 | 63 | 68 | 0 | 1114 |
| dfiles | 3078 | 837 | 105 | 216 | 1328 | 304 | 93 | 0 | 0 | 0 | 5961 |
| gfiles | 1361 | 497 | 12 | 127 | 437 | 185 | 20 | 2 | 0 | 0 | 2641 |
| hfiles | 2146 | 411 | 138 | 298 | 961 | 210 | 219 | 0 | 0 | 0 | 4383 |
| lfiles | 135 | 1 | 8 | 26 | 68 | 3 | 20 | 0 | 0 | 0 | 261 |
| mfiles | 424 | 354 | 6 | 39 | 164 | 68 | 10 | 0 | 0 | 0 | 1065 |
| nfiles | 2110 | 463 | 53 | 145 | 685 | 205 | 54 | 0 | 0 | 0 | 3715 |
| pfiles | 6292 | 943 | 236 | 941 | 2646 | 300 | 436 | 0 | 0 | 0 | 11794 |
| qfiles | 39 | 45 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 85 |
| rfiles | 2369 | 602 | 22 | 234 | 838 | 188 | 51 | 0 | 0 | 0 | 4304 |
| ufiles | 4533 | 336 | 101 | 709 | 2590 | 249 | 279 | 0 | 0 | 0 | 8797 |
| wfiles | 311 | 36 | 15 | 36 | 136 | 7 | 28 | 0 | 0 | 0 | 569 |
| xfiles | 500 | 55 | 30 | 45 | 369 | 28 | 36 | 32 | 422 | 4 | 1521 |
| yfiles | 45 | 0 | 6 | 14 | 18 | 1 | 11 | 0 | 0 | 0 | 95 |
| total | 24684 | 4832 | 869 | 3060 | 10841 | 1902 | 1445 | 97 | 490 | 4 | 48224 |
| % | 51 | 10 | 2 | 6 | 22 | 4 | 3 | 0 | 1 | 0 | |

Table 1. Instruction Frequencies for AV8B Version 5

| Name | load | arith | logic | comp | jump | shift | misc | exec | chain | stack |
|---|---|---|---|---|---|---|---|---|---|---|
| afiles | 56 | 21 | 0 | 1 | 12 | 11 | 0 | 0 | 0 | 0 |
| bfiles | 42 | 7 | 6 | 10 | 19 | 5 | 11 | 0 | 0 | 0 |
| cfiles | 42 | 6 | 5 | 7 | 24 | 3 | 2 | 6 | 6 | 0 |
| dfiles | 52 | 14 | 2 | 4 | 22 | 5 | 2 | 0 | 0 | 0 |
| gfiles | 52 | 19 | 0 | 5 | 17 | 7 | 1 | 0 | 0 | 0 |
| hfiles | 49 | 9 | 3 | 7 | 22 | 5 | 5 | 0 | 0 | 0 |
| lfiles | 52 | 0 | 3 | 10 | 26 | 1 | 8 | 0 | 0 | 0 |
| mfiles | 40 | 33 | 1 | 4 | 15 | 6 | 1 | 0 | 0 | 0 |
| nfiles | 57 | 12 | 1 | 4 | 18 | 6 | 1 | 0 | 0 | 0 |
| pfiles | 53 | 8 | 2 | 8 | 22 | 3 | 4 | 0 | 0 | 0 |
| qfiles | 46 | 53 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| rfiles | 55 | 14 | 1 | 5 | 19 | 4 | 1 | 0 | 0 | 0 |
| ufiles | 52 | 4 | 1 | 8 | 29 | 3 | 3 | 0 | 0 | 0 |
| wfiles | 55 | 6 | 3 | 6 | 24 | 1 | 5 | 0 | 0 | 0 |
| xfiles | 33 | 4 | 2 | 3 | 24 | 2 | 2 | 2 | 28 | 0 |
| yfiles | 47 | 0 | 6 | 15 | 19 | 1 | 12 | 0 | 0 | 0 |

Table 2. Percentages for AV8B Version 5 Instructions.

| Name | load | arith | logic | comp | jump | shift | misc | exec | chain | stack | total |
|---|---|---|---|---|---|---|---|---|---|---|---|
| afiles | 238 | 89 | 1 | 3 | 49 | 46 | 0 | 0 | 0 | 0 | 426 |
| bfiles | 660 | 101 | 85 | 151 | 300 | 81 | 172 | 0 | 0 | 0 | 1550 |
| cfiles | 514 | 63 | 53 | 85 | 279 | 30 | 17 | 64 | 68 | 0 | 1173 |
| dfiles | 3098 | 845 | 105 | 215 | 1330 | 307 | 93 | 0 | 0 | 0 | 5993 |
| gfiles | 2216 | 938 | 28 | 216 | 676 | 340 | 23 | 2 | 0 | 0 | 4439 |
| hfiles | 2382 | 448 | 160 | 331 | 1079 | 223 | 247 | 0 | 0 | 0 | 4870 |
| lfiles | 129 | 2 | 8 | 23 | 62 | 3 | 19 | 0 | 0 | 0 | 246 |
| mfiles | 424 | 354 | 6 | 39 | 164 | 68 | 10 | 0 | 0 | 0 | 1065 |
| nfiles | 2315 | 519 | 53 | 157 | 737 | 209 | 54 | 0 | 0 | 0 | 4044 |
| pfiles | 6722 | 1035 | 253 | 1048 | 2866 | 324 | 468 | 0 | 0 | 0 | 12716 |
| qfiles | 39 | 45 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 85 |
| rfiles | 2496 | 620 | 30 | 248 | 885 | 200 | 52 | 0 | 0 | 0 | 4531 |
| ufiles | 5025 | 393 | 121 | 778 | 2846 | 302 | 285 | 0 | 0 | 0 | 9750 |
| wfiles | 448 | 54 | 18 | 45 | 196 | 3 | 37 | 0 | 0 | 0 | 801 |
| xfiles | 552 | 59 | 30 | 48 | 386 | 28 | 37 | 35 | 443 | 4 | 1622 |
| yfiles | 45 | 0 | 6 | 14 | 18 | 1 | 11 | 0 | 0 | 0 | 95 |
| total | 27303 | 5565 | 957 | 3401 | 11873 | 2165 | 1526 | 101 | 511 | 4 | 53406 |
| % | 51 | 10 | 2 | 6 | 22 | 4 | 3 | 0 | 1 | 0 | |

Table 3. Instruction Frequencies for AV8B Version 7

| Name | load | arith | logic | comp | jump | shift | misc | exec | chain | stack |
|---|---|---|---|---|---|---|---|---|---|---|
| afiles | 56 | 21 | 0 | 1 | 12 | 11 | 0 | 0 | 0 | 0 |
| bfiles | 43 | 7 | 5 | 10 | 19 | 5 | 11 | 0 | 0 | 0 |
| cfiles | 44 | 5 | 5 | 7 | 24 | 3 | 1 | 5 | 6 | 0 |
| dfiles | 52 | 14 | 2 | 4 | 22 | 5 | 2 | 0 | 0 | 0 |
| gfiles | 50 | 21 | 1 | 5 | 15 | 8 | 1 | 0 | 0 | 0 |
| hfiles | 49 | 9 | 3 | 7 | 22 | 5 | 5 | 0 | 0 | 0 |
| lfiles | 52 | 1 | 3 | 9 | 25 | 1 | 8 | 0 | 0 | 0 |
| mfiles | 40 | 33 | 1 | 4 | 15 | 6 | 1 | 0 | 0 | 0 |
| nfiles | 57 | 13 | 1 | 4 | 18 | 5 | 1 | 0 | 0 | 0 |
| pfiles | 53 | 8 | 2 | 8 | 23 | 3 | 4 | 0 | 0 | 0 |
| qfiles | 46 | 53 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| rfiles | 55 | 14 | 1 | 5 | 20 | 4 | 1 | 0 | 0 | 0 |
| ufiles | 52 | 4 | 1 | 8 | 29 | 3 | 3 | 0 | 0 | 0 |
| wfiles | 56 | 7 | 2 | 6 | 24 | 0 | 5 | 0 | 0 | 0 |
| xfiles | 34 | 4 | 2 | 3 | 24 | 2 | 2 | 2 | 27 | 0 |
| yfiles | 47 | 0 | 6 | 15 | 19 | 1 | 12 | 0 | 0 | 0 |

Table 4 Percentages for AV8B Version 7 Instructions.

| Name | load | arith | logic | comp | jump | shift | misc | exec | chain | stack | total |
|---|---|---|---|---|---|---|---|---|---|---|---|
| afiles | 238 | 89 | 1 | 3 | 49 | 46 | 0 | 0 | 0 | 0 | 426 |
| bfiles | 693 | 105 | 85 | 156 | 324 | 80 | 179 | 0 | 0 | 0 | 1622 |
| cfiles | 461 | 57 | 52 | 66 | 231 | 27 | 19 | 65 | 50 | 0 | 1028 |
| dfiles | 3653 | 986 | 142 | 254 | 1585 | 350 | 115 | 0 | 0 | 0 | 7085 |
| gfiles | 1466 | 501 | 15 | 135 | 449 | 188 | 24 | 2 | 0 | 0 | 2780 |
| hfiles | 2516 | 434 | 185 | 317 | 1073 | 236 | 250 | 0 | 0 | 0 | 5011 |
| jfiles | 30 | 1 | 0 | 0 | 6 | 0 | 0 | 4 | 0 | 0 | 41 |
| lfiles | 128 | 2 | 8 | 23 | 62 | 3 | 19 | 0 | 0 | 0 | 245 |
| mfiles | 263 | 168 | 8 | 32 | 104 | 19 | 2 | 0 | 0 | 0 | 596 |
| nfiles | 2391 | 533 | 56 | 162 | 783 | 222 | 65 | 0 | 0 | 0 | 4212 |
| pfiles | 9324 | 897 | 365 | 1355 | 4140 | 405 | 610 | 1 | 0 | 0 | 17097 |
| qfiles | 151 | 157 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 309 |
| rfiles | 2593 | 605 | 31 | 259 | 925 | 203 | 50 | 0 | 0 | 0 | 4666 |
| sfiles | 124 | 11 | 6 | 5 | 56 | 3 | 4 | 29 | 0 | 0 | 238 |
| tfiles | 291 | 48 | 3 | 50 | 93 | 8 | 16 | 0 | 0 | 0 | 509 |
| ufiles | 6304 | 434 | 165 | 892 | 3363 | 329 | 314 | 0 | 0 | 0 | 11801 |
| wfiles | 465 | 52 | 27 | 44 | 210 | 1 | 35 | 0 | 0 | 0 | 834 |
| xfiles | 580 | 60 | 31 | 61 | 415 | 30 | 46 | 52 | 503 | 4 | 1782 |
| yfiles | 45 | 0 | 6 | 14 | 18 | 1 | 11 | 0 | 0 | 0 | 95 |
| total | 31716 | 5140 | 1186 | 3828 | 13886 | 2151 | 1760 | 153 | 553 | 4 | 60377 |
| % | 53 | 9 | 2 | 6 | 23 | 4 | 3 | 0 | 1 | 0 | |

Table 5. Instruction Frequencies for AV8B Version 6.

| Name | load | arith | logic | comp | jump | shift | misc | exec | chain | stack |
|---|---|---|---|---|---|---|---|---|---|---|
| afiles | 56 | 21 | 0 | 1 | 12 | 11 | 0 | 0 | 0 | 0 |
| bfiles | 43 | 6 | 5 | 10 | 20 | 5 | 11 | 0 | 0 | 0 |
| cfiles | 45 | 6 | 5 | 6 | 22 | 3 | 2 | 6 | 5 | 0 |
| dfiles | 52 | 14 | 2 | 4 | 22 | 5 | 2 | 0 | 0 | 0 |
| gfiles | 53 | 18 | 1 | 5 | 16 | 7 | 1 | 0 | 0 | 0 |
| hfiles | 50 | 9 | 4 | 6 | 21 | 5 | 5 | 0 | 0 | 0 |
| jfiles | 73 | 2 | 0 | 0 | 15 | 0 | 0 | 1 | 0 | 0 |
| lfiles | 52 | 1 | 3 | 9 | 25 | 1 | 8 | 0 | 0 | 0 |
| mfiles | 44 | 28 | 1 | 5 | 17 | 3 | 0 | 0 | 0 | 0 |
| nfiles | 57 | 13 | 1 | 4 | 19 | 5 | 2 | 0 | 0 | 0 |
| pfiles | 55 | 5 | 2 | 8 | 24 | 2 | 4 | 0 | 0 | 0 |
| qfiles | 49 | 51 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| rfiles | 56 | 13 | 1 | 6 | 20 | 4 | 1 | 0 | 0 | 0 |
| sfiles | 52 | 5 | 3 | 2 | 24 | 1 | 2 | 12 | 0 | 0 |
| tfiles | 57 | 9 | 1 | 10 | 18 | 2 | 3 | 0 | 0 | 0 |
| ufiles | 53 | 4 | 1 | 8 | 28 | 3 | 3 | 0 | 0 | 0 |
| wfiles | 56 | 6 | 3 | 5 | 25 | 0 | 4 | 0 | 0 | 0 |
| xfiles | 33 | 3 | 2 | 3 | 23 | 2 | 3 | 3 | 28 | 0 |
| yfiles | 47 | 0 | 6 | 15 | 19 | 1 | 12 | 0 | 0 | 0 |

Table 6. Percentages for AV8B Version 6 Instructions.

| Name | load | arith | logic | comp | jump | shift | misc | exec | chain | stack | total |
|---|---|---|---|---|---|---|---|---|---|---|---|
| afiles | 3889 | 1177 | 199 | 519 | 1577 | 442 | 165 | 0 | 0 | 0 | 7968 |
| bfiles | 1730 | 364 | 222 | 501 | 949 | 193 | 506 | 0 | 0 | 0 | 4465 |
| cfiles | 571 | 74 | 46 | 115 | 324 | 36 | 19 | 79 | 77 | 0 | 1341 |
| dfiles | 1655 | 83 | 95 | 294 | 959 | 33 | 162 | 0 | 397 | 0 | 3678 |
| efiles | 1025 | 222 | 31 | 240 | 450 | 68 | 69 | 0 | 0 | 0 | 2105 |
| gfiles | 5853 | 1424 | 155 | 688 | 2469 | 732 | 245 | 0 | 0 | 0 | 11566 |
| hfiles | 6444 | 746 | 253 | 1055 | 3333 | 410 | 356 | 0 | 0 | 0 | 12597 |
| lfiles | 1266 | 148 | 142 | 259 | 560 | 95 | 146 | 0 | 0 | 0 | 2616 |
| mfiles | 398 | 381 | 12 | 26 | 146 | 81 | 11 | 0 | 0 | 0 | 1055 |
| nfiles | 2833 | 602 | 78 | 274 | 1083 | 270 | 120 | 0 | 0 | 0 | 5260 |
| pfiles | 3454 | 381 | 179 | 821 | 1847 | 85 | 468 | 0 | 0 | 0 | 7235 |
| qfiles | 44 | 8 | 10 | 4 | 18 | 2 | 0 | 0 | 0 | 0 | 86 |
| sfiles | 2087 | 188 | 200 | 316 | 1008 | 134 | 212 | 0 | 0 | 0 | 4145 |
| tfiles | 6509 | 742 | 360 | 1555 | 3608 | 458 | 860 | 0 | 0 | 0 | 14092 |
| ufiles | 3311 | 492 | 90 | 530 | 1757 | 172 | 110 | 0 | 0 | 0 | 6462 |
| xfiles | 1022 | 81 | 73 | 109 | 652 | 45 | 79 | 99 | 530 | 5 | 2695 |
| zfiles | 1202 | 142 | 119 | 175 | 454 | 125 | 80 | 2 | 16 | 0 | 2315 |
| total | 43293 | 7255 | 2264 | 7481 | 21194 | 3381 | 3608 | 180 | 1020 | 5 | 89681 |
| % | 48 | 8 | 3 | 8 | 24 | 4 | 4 | 0 | 1 | 0 | |

Table 7. Instruction Frequencies for F18 Version 87X

| Name | load | arith | logic | comp | jump | shift | misc | exec | chain | stack |
|---|---|---|---|---|---|---|---|---|---|---|
| afiles | 49 | 15 | 2 | 7 | 20 | 6 | 2 | 0 | 0 | 0 |
| bfiles | 39 | 8 | 5 | 11 | 21 | 4 | 11 | 0 | 0 | 0 |
| cfiles | 43 | 6 | 3 | 9 | 24 | 3 | 1 | 6 | 6 | 0 |
| dfiles | 45 | 2 | 3 | 8 | 26 | 1 | 4 | 0 | 11 | 0 |
| efiles | 49 | 11 | 1 | 11 | 21 | 3 | 3 | 0 | 0 | 0 |
| gfiles | 51 | 12 | 1 | 6 | 21 | 6 | 2 | 0 | 0 | 0 |
| hfiles | 51 | 6 | 2 | 8 | 26 | 3 | 3 | 0 | 0 | 0 |
| lfiles | 48 | 6 | 5 | 10 | 21 | 4 | 6 | 0 | 0 | 0 |
| mfiles | 38 | 36 | 1 | 2 | 14 | 8 | 1 | 0 | 0 | 0 |
| nfiles | 54 | 11 | 1 | 5 | 21 | 5 | 2 | 0 | 0 | 0 |
| pfiles | 48 | 5 | 2 | 11 | 26 | 1 | 6 | 0 | 0 | 0 |
| qfiles | 51 | 9 | 12 | 5 | 21 | 2 | 0 | 0 | 0 | 0 |
| sfiles | 50 | 5 | 5 | 8 | 24 | 3 | 5 | 0 | 0 | 0 |
| tfiles | 46 | 5 | 3 | 11 | 26 | 3 | 6 | 0 | 0 | 0 |
| ufiles | 51 | 8 | 1 | 8 | 27 | 3 | 2 | 0 | 0 | 0 |
| xfiles | 38 | 3 | 3 | 4 | 24 | 2 | 3 | 4 | 20 | 0 |
| zfiles | 52 | 6 | 5 | 8 | 20 | 5 | 3 | 0 | 1 | 0 |

Table 8. Percentage of F18 Version 87X Instructions.

| Name | load | arith | logic | comp | jump | shift | misc | exec | chain | stack | total |
|---|---|---|---|---|---|---|---|---|---|---|---|
| afiles | 4146 | 1211 | 211 | 584 | 1733 | 464 | 188 | 0 | 0 | 0 | 8537 |
| bfiles | 1807 | 366 | 223 | 528 | 1000 | 196 | 517 | 0 | 0 | 0 | 4637 |
| cfiles | 571 | 74 | 46 | 115 | 326 | 36 | 19 | 79 | 77 | 0 | 1343 |
| dfiles | 1675 | 83 | 97 | 305 | 977 | 33 | 163 | 0 | 397 | 0 | 3730 |
| efiles | 1075 | 229 | 35 | 256 | 487 | 72 | 76 | 0 | 0 | 0 | 2230 |
| gfiles | 6227 | 1522 | 164 | 730 | 2586 | 800 | 254 | 0 | 0 | 0 | 12283 |
| hfiles | 5651 | 693 | 222 | 920 | 2913 | 374 | 314 | 0 | 0 | 0 | 11087 |
| lfiles | 1279 | 149 | 142 | 260 | 562 | 95 | 146 | 0 | 0 | 0 | 2633 |
| mfiles | 398 | 381 | 12 | 26 | 146 | 81 | 11 | 0 | 0 | 0 | 1055 |
| nfiles | 3128 | 654 | 75 | 321 | 1239 | 280 | 117 | 0 | 0 | 0 | 5814 |
| pfiles | 3507 | 380 | 180 | 830 | 1886 | 87 | 472 | 0 | 0 | 0 | 7342 |
| qfiles | 44 | 8 | 10 | 4 | 18 | 2 | 0 | 0 | 0 | 0 | 86 |
| sfiles | 2147 | 190 | 201 | 322 | 1030 | 135 | 224 | 0 | 0 | 0 | 4249 |
| tfiles | 7361 | 896 | 414 | 1738 | 4024 | 493 | 997 | 0 | 0 | 0 | 15923 |
| ufiles | 3391 | 512 | 98 | 554 | 1804 | 175 | 113 | 0 | 0 | 0 | 6647 |
| xfiles | 1039 | 82 | 76 | 112 | 663 | 47 | 78 | 104 | 530 | 5 | 2736 |
| zfiles | 1213 | 143 | 120 | 177 | 456 | 126 | 80 | 2 | 16 | 0 | 2333 |
| total | 44659 | 7573 | 2326 | 7782 | 21850 | 3496 | 3769 | 185 | 1020 | 5 | 92665 |
| % | 48 | 8 | 3 | 8 | 24 | 4 | 4 | 0 | 1 | 0 | |

Table 9. Instruction Frequencies for F18 Version 89A.

| Name | load | arith | logic | comp | jump | shift | misc | exec | chain | stack |
|---|---|---|---|---|---|---|---|---|---|---|
| afiles | 49 | 14 | 2 | 7 | 20 | 5 | 2 | 0 | 0 | 0 |
| bfiles | 39 | 8 | 5 | 11 | 22 | 4 | 11 | 0 | 0 | 0 |
| cfiles | 43 | 6 | 3 | 9 | 24 | 3 | 1 | 6 | 6 | 0 |
| dfiles | 45 | 2 | 3 | 8 | 26 | 1 | 4 | 0 | 11 | 0 |
| efiles | 48 | 10 | 1 | 11 | 22 | 3 | 3 | 0 | 0 | 0 |
| gfiles | 51 | 12 | 1 | 6 | 21 | 7 | 2 | 0 | 0 | 0 |
| hfiles | 51 | 6 | 2 | 8 | 26 | 3 | 3 | 0 | 0 | 0 |
| lfiles | 49 | 6 | 5 | 10 | 21 | 4 | 6 | 0 | 0 | 0 |
| mfiles | 38 | 36 | 1 | 2 | 14 | 8 | 1 | 0 | 0 | 0 |
| nfiles | 54 | 11 | 1 | 6 | 21 | 5 | 2 | 0 | 0 | 0 |
| pfiles | 48 | 5 | 2 | 11 | 26 | 1 | 6 | 0 | 0 | 0 |
| qfiles | 51 | 9 | 12 | 5 | 21 | 2 | 0 | 0 | 0 | 0 |
| sfiles | 51 | 4 | 5 | 8 | 24 | 3 | 5 | 0 | 0 | 0 |
| tfiles | 46 | 6 | 3 | 11 | 25 | 3 | 6 | 0 | 0 | 0 |
| ufiles | 51 | 8 | 1 | 8 | 27 | 3 | 2 | 0 | 0 | 0 |
| xfiles | 38 | 3 | 3 | 4 | 24 | 2 | 3 | 4 | 19 | 0 |
| zfiles | 52 | 6 | 5 | 8 | 20 | 5 | 3 | 0 | 1 | 0 |

Table 10. Percentages of F18 Version 89A Instructions.

## INFORMATION THEORY BASED SOFTWARE COMPLEXITY MEASURE

The major programming task performed on the avionics software is maintenance. Since up to half of maintenance programmer's time is spent studying the code and related documentation, the understandability of the software is a major complexity factor. Hence we believe that a maintenance programmer will have a more difficult time understanding a program with unfamiliar assembly language instructions than a program containing only familiar instructions. If we assume that a programmer is more familiar with commonly used instruction that with rarely used instructions, then the distribution of instructions provides a method of classifying instructions as familiar or unfamiliar. For our data the frequently used instructions are the load/store, jump, arithmetic, etc. and the rarely used instructions are the command/chain, executive mode, and stack/queue instructions since they are used 1% or less of the time.

Berlinger [1 ] developed an information theory complexity measure is based on the the frequency of tokens in the program. His complexity measure M is defined by

$$M = -\Sigma f_i \log p_i$$

where $f_i$ is the frequency of the ith token and $p_i$ is the probability of the ith token.

M is sensitive to the frequency and probability of tokens. M will have a low value if many high probability (e.g. familiar) tokens are used. On the other hand M will have a high value if there are many low probability (e.g. unfamiliar) tokens. An analogy is the influence of word familiarity on the difficulty of understanding or reading written text. Generally text comprehension and reading is easier and faster if it contains many familiar words. Unfamiliar words slow down the reading and comprehension task.

Our information based complexity measure M' is based on a formula by Berlinger. We define M' by

$$M' = M/(\text{ number of instructions})$$

Our tokens in Berlinger's M are the 10 classes of assembly language instructions. Their probabilities are the number of times they occur divided by the total number of instructions. We divided M by the number of instructions because we found that the instruction frequencies dominated the value of M. (Our smallest files contained less than 100 instructions and the largest over 10,000.) We wanted our metric to better reflect the distribution of the tokens, especially

the rarely occurring tokens Hence we changed Berlinger's complexity measure M so that it computed a weighted mean.

Table 11 gives the frequencies of the classes of assembly language instructions for the 5 data sets, the probability of each class and the logarithm (base 2) of each probability The last column is the weight of each instruction class. Note that the load/store instructions have a weight of about one while the executive mode, chain/command, and stack/queue instructions have weights above 6.5. Table 12 gives the M' values for each of the files and for each file in the 5 data sets. The M' value for all the files combined is 2.1348.

From Table 12 we see that the only files to exceed 3.0 are the Xfiles in all five data sets. The next highest M' values are the Cfiles which are above 2.7 for all five data sets. Both of these sets of files have high M' values because they contain a high proportion of the executive mode and chain/command instructions. Note that only the Xfiles contain the few stack/queue instructions which have a weight of over 13. But merely containing executive mode or chain/command instructions does not guarantee a high M' value. For example, the Gfiles and Jfiles (AV8B) and the Zfiles (F18) all contain chain/command and executive mode instructions, but their M' values are close to or less than the mean. For the AV8B, the X-, C-, B-, and M-files had M' values greater than 2.4 while for the F18 the X-, C-, B-, M- and D- files had values greater than 2.4..

## PROGRAMMER DATA

Programmers responsible for maintaining the AV8B software were asked to rate the difficulty of the software. The list of what they felt were the most complex files included the mathematical (M files), executive (X files), and Bit and Self-test files (B files). They felt that parameterization, executive instructions used, and familiarity with the domain made these files complex. The X files are rated most difficult by our metric and the B and M files are rated third and fourth most difficult.

The list of files provided by the programmers also included a few files the other function classes. These files were not rated as unusually complex by our metric; however, these files were among the largest which suggests that the programmers considered program size an important difficulty factor.

## CONCLUSION

The analysis of the distribution of assembly language instructions showed the expected, but it also produced some surprises. Distribution of the instruction classes over the modules was generally uniform. Nearly three-fourths of the instructions were load, store, or jump instructions. The biggest surprise was that that nearly one-half of the 320 op codes were not used. We suspect that for

assembly languages the larger the number of different instruction, the larger the proportion of unused instructions.

The basic premise for our information theory software complexity metric is that programs with familiar instructions are much easier to understand and work with than programs with unfamiliar instructions. Because of such factors as programmer preferences and application domain, the sets of familiar and unfamiliar instructions will vary between programmers. We used frequency of instruction use as an approximation for instruction familiarity in our metric. This seems to reasonable because it is flexible and easy to compute.

We grouped the instructions into related classes and then computed the probabilities for each class. An alternative would be to compute the probability for each instruction. We feel that the alternative will give too much weight to a rarely used instructions, should be investigated.

We chose the average information content per instruction for our metric rather than the sum of information content for the instructions. We felt that it was important to know whether a large information value was because of a large number of instructions or because of infrequently used instructions. It may be possible to develop a metric that incorporates both size and information content.

The metric values agreed reasonably well with the subjective judgements of the maintenance programmers except for large programs. More data needs to be collected from different applications and for different assembly languages to validate the metric.

| | All Files | Fraction of total | Log(Fraction) |
|---|---|---|---|
| load/store | 171,655 | 0.4984 | -1.004 |
| Arithmetic | 30,365 | 0.0881 | -3.503 |
| Logical | 7,602 | 0.0220 | -5.501 |
| Compare | 25,552 | 0.0742 | -3.752 |
| Jump | 79,644 | 0.2312 | -2.112 |
| Shift | 13,095 | 0.0380 | -4.716 |
| Miscellaneous | 12,108 | 0.0351 | -4.829 |
| Executive Mode | 716 | 0.0020 | -8.909 |
| Chain / Command | 3,594 | 0.0104 | -6.582 |
| Stack/Queue | 22 | 0.000064 | -13.93 |
| TOTAL | 344,353 | | |

Table 11. Total, fraction and log(fraction) for each instruction class.

| FILES | V5 | V7 | V6 | 87X | 89A |
|---|---|---|---|---|---|
| Afiles | 2.0846 | 2.0846 | 2.0846 | 2.1692 | 2.1689 |
| Bfiles | 2.5356 | 2.5144 | 2.4926 | 2.5695 | 2.5531 |
| Cfiles | 2.7653 | 2.7094 | 2.7351 | 2.7397 | 2.7387 |
| Dfiles | 2.0298 | 2.0295 | 2.0341 | 2.4893 | 2.4855 |
| Efiles | | | | 2.1296 | 2.1392 |
| Gfiles | 2.1055 | 2.1709 | 2.0811 | 2.0882 | 2.0915 |
| Hfiles | 2.1790 | 2.1782 | 2.1636 | 1.9949 | 2.0033 |
| Jfiles | | | 1.9987 | | |
| Lfiles | 2.0499 | 2.0478 | 2.0520 | 2.2474 | 2.2422 |
| Mfiles | 2.4045 | 2.4045 | 2.2412 | 2.5039 | 2.5039 |
| Nfiles | 1.9519 | 1.9355 | 1.9467 | 2.0061 | 1.9870 |
| Pfiles | 1.9978 | 2.0088 | 1.9424 | 2.1329 | 2.1291 |
| Qfiles | 2.3723 | 2.3723 | 2.2864 | 2.2057 | 2.2057 |
| Rfiles | 1.9495 | 1.9506 | 1.9329 | | |
| Sfiles | | | 2.6259 | 2.1292 | 2.1252 |
| Tfiles | | | 1.9175 | 2.1918 | 2.1963 |
| Ufiles | 1.9255 | 1.9303 | 1.8878 | 1.9478 | 1.9566 |
| Wfiles | 1.9536 | 1.8899 | 1.8947 | | |
| Xfiles | 3.4209 | 3.4006 | 3.5142 | 3.1659 | 3.1625 |
| Yfiles | 2.3853 | 2.3853 | 2.3853 | | |
| Zfiles | | | | 2.1918 | 2.1905 |
| File Total | 2.0933 | 2.0957 | 2.0502 | 2.1850 | 2.1870 |

Table 12. Information Theory Metric for files.

## References


1. E. Berlinger, An Information Theory Based Complexity Measure, *Proceedings of the 1980 National Computer Conference*. pp. 773-779.

2. S. Conte, H. Dunsmore, and V. Shen, *Software Engineering Metrics and Models*. Benjamin/Cummings, Menlo Park, California, 1986.

3. D. H. Howes and R. L. Solomon, Visual Duration Threshold as a Function of Word-Probability, *Journal of Experimental Psychology* ,Vol. 41 (June 1951), pp. 401-410.

4. G. Klare, *The Measurement of Readability*. Iowa State University Press, Ames, Iowa, 1963.

5. R. P. O'Reilly and J. E. Walker, An Analysis of Reading Rates in College Students, *Reading Research and Instruction*, Vol. 29 No. 2 (Winter 1990), pp. 1-11.