IBM High Level Assembler Toolkit Feature: Program Understanding Tool (ASMPUT)

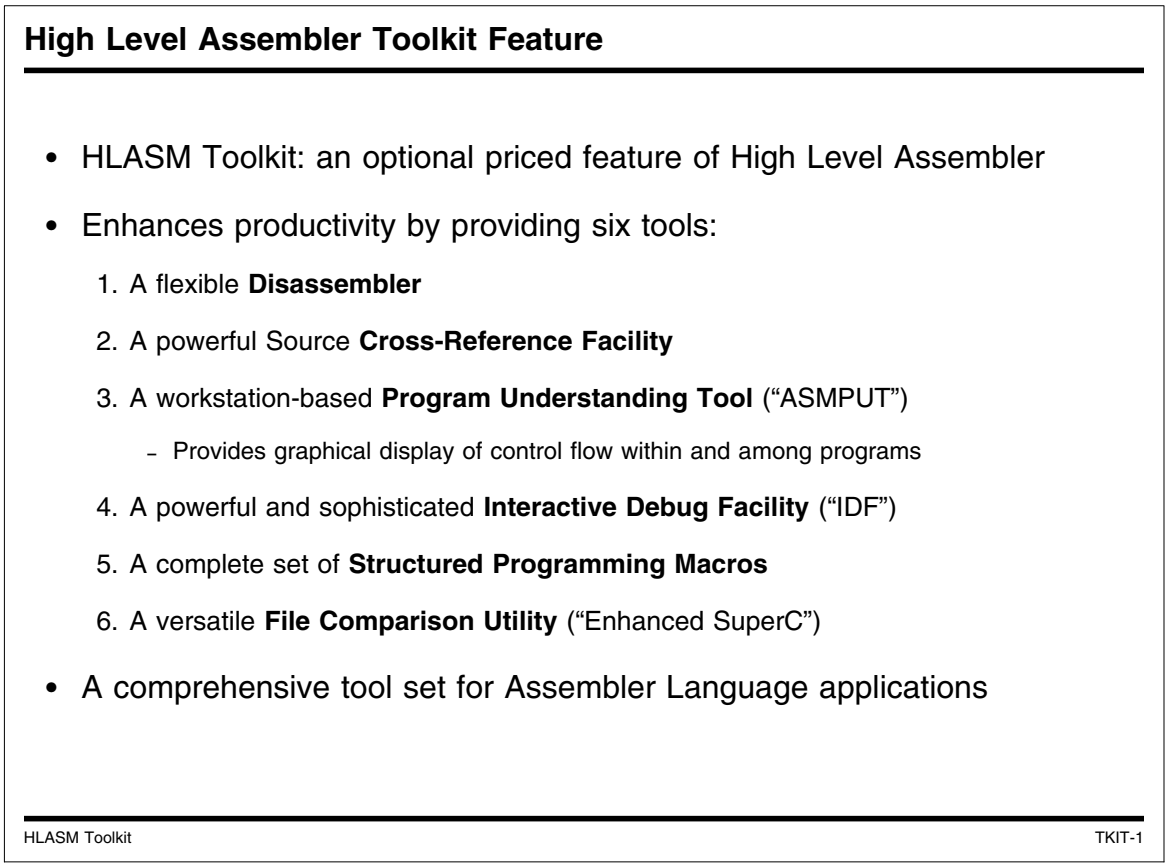
IBM High Level Assembler Rel 3
Toolkit Feature
Program Understanding Tool (ASMPUT)
VM/VSE Technical Conference
Orlando 2000

Clive Nealon

cliven@au1.ibm.com

May, 2000

Table of Contents



High Level Assembler Toolkit Feature

The High Level Assembler Toolkit Feature is an optional, separately priced feature of IBM High Level Assembler for MVS & VM & VSE (HLASM). It provides a powerful and flexible set of six tools to improve application recovery and development on OS/390*, MVS/ESA*, VM/ESA*, and VSE/ESA* systems.

Together, these tools provide a powerful set of capabilities to speed application development, diagnosis, and recovery.

This presentation provides an overview of the features and usage of the Assembler Program Understanding Tool.

The Toolkit Feature's components can be used independently of HLASM. However, the most productive uses of most of the Toolkit Feature's components rely on SYSADATA files produced by High Level Assembler.

For more information about ordering the High Level Assembler Toolkit Feature, refer to Software Announcement 295-498, dated December 12, 1995.

Hardware Requirements

The High Level Assembler Toolkit Feature requires the same hardware environments as IBM High Level Assembler for MVS & VM & VSE Version 1 Release 3. Requirements for 24-bit Virtual Storage are:

- Disassembler: 250K bytes
- IDF: 600K bytes
- XREF: depends on number and sizes of modules being scanned
- SuperC: depends on number and sizes of modules being scanned
- ...plus working storage (depending on the application)

The Program Understanding Tool (ASMPUT) component of the High Level Assembler Toolkit Feature requires a workstation capable of running OS/2, Windows 95, Windows 98, or Windows NT with a minimum of 16 MB memory (32 MB recommended) and 10 MB of disk drive space, plus a host-sytem connection or some other means of transferring SYSADATA files to the workstation for analysis.

Toolkit Software Requirements

- IDF
  - On VSE, requires VSE Version 2.2 or later
- ASMPUT
  - APAR PQ26063
  - One of:
    - OS/2 Version 4 (8H1425) with fixpack 8 or later
    - Windows 95
    - Windows 98
    - Windows NT Version 4.0 with Service Pack 3 or later (Intel)

      Note: On Windows platforms, Microsoft Internet Explorer V4 or later required.
  - HLASM R3 SYSADATA files only
    - Do not create with the XOBJECT or GOFF options.
  - A means to transfer files from the host to the workstation.

HLASM Toolkit TKIT-2

Toolkit Software Requirements

The High Level Assembler Toolkit Feature operates in all environments where IBM High Level Assembler for MVS & VM & VSE Version 1 Release 3 operates.

On VSE, the Interactive Debug Facility requires VSE Version 2.2 or later.

ASMPUT

APAR PQ26063 provides an updated version of the Windows ASMPUT. This release has far greater functionality than the previous release. The OS/2 version provides similar functionality to the previous release, but uses the enhanced interface provided with the Windows version, making for much cleaner use.

The help for the Windows versions is provided in the latest HTML Help format. This requires Internet Explorer V4 (or later) to run. ASMPUT runs if IE4 is not provided, but you won't be able to access the online help.

If you have ADATA files generated by HLASM R2 and want to continue to use them, you should retain your copy of the OS/2-based ASMPUT shipped with HLASM R2.

A recommended host-connection software package is eNetwork Personal Communications Version 4.2.1 (8H8735), which supports OS/2 and Windows.

HLASM Toolkit Publications

GC26-8710 *Toolkit Feature User's Guide*

Reference and usage information for all Toolkit Feature components except IDF.

GC26-8709 *Toolkit Feature Interactive Debug Facility User's Guide*

The reference document for IDF.

GC26-8712 *Toolkit Feature Interactive Debug Facility Reference Summary*

Quick-reference summary for experienced IDF users.

GC26-8711 *Toolkit Feature Installation and Customization Guide*

Information needed to install all Toolkit Feature components.

www.ibm.com/software/ad/hlasm The HLASM web site, including the ASMPUT demo, and the 30-day trial version (Windows 95 or up).

HLASM Toolkit TKIT-3

HLASM Toolkit Publications

The four publications for the High Level Assembler Toolkit Feature are:

GC26-8710 *Toolkit Feature User's Guide*

Reference and usage information for the Disassembler, the Cross-Reference Facility, the Program Understanding Tool, the Enhanced

SuperC File Comparison Utility, and the Structured Programming Macros.

GC26-8709 *Toolkit Feature Interactive Debug Facility User's Guide*

The document that describes all IDF facilities, commands, windows and messages in detail.

GC26-8712 *Toolkit Feature Interactive Debug Facility Reference Summary*

A quick reference summary, with syntax for all commands and a list of all the options. This booklet is intended for experienced IDF users.

GC26-8711 *Toolkit Feature Installation and Customization Guide*

Information needed to install all Toolkit Feature components.

The HLASM web site holds all these documents in PDF and HTML format, and the other HLASM publications. As well, the web site holds the ASMPUT slide-show demonstration, and a 30-day trial version, which runs under Windows 95 (or later). The web site also has two IDF slide-show demos.

HLASM Toolkit Program Understanding Tool (ASMPUT)

- Detailed analysis of Assembler Language programs
  - Supports latest processor-family instructions
  - Creates annotated listings
  - Displays graphic control flow for single programs and “linked” modules
- Assemble programs with ADATA option
  - Download SYSADATA file (in binary) to workstation `*.XAA` files
- ASMPUT analyzes the SYSADATA (`.XAA`) files
  - Creates component lists, simulated listing, graphs, external linkages
- Graph displays many levels of detail, with zoom capability
  - Inter-program relationships
  - Major program structures
  - Full details of internal control flows
- Online Help, slide-show demonstration
- Installed from downloaded host files (not diskettes)

HLASM Toolkit TKIT-4

HLASM Toolkit Program Understanding Tool (ASMPUT)

ASMPUT helps you analyze and extract information about Assembler Language applications, using a graphical user interface to display graphical and textual views of an application's structure.

ASMPUT extracts application analysis information from the SYSADATA file generated during host assembly by HLASM. ADATA files are downloaded to the workstation for analysis.

Notes:

1. The High Level Assembler Toolkit Feature Release 3 ASMPUT requires ADATA files generated by HLASM Release 3.
2. Do not use the GOFF or XOBJECT option when you are creating SYSADATA files for ASMPUT analysis.

ASMPUT displays views of selected programs and modules including:

- The source code (listing for one program or module)
- The control flow graph (one graph for all programs and modules)
- The overview (a small window showing all of the control flow graph)

These views provide complete high, medium, and low level information about Assembler Language applications.

- At the highest level, you can discover the relationships among programs and modules within an application.
- At a mid level, you can see the the calling structures among programs within a module, including routines external to a program.
- At the lowest level, you can examine details of internal control flows within each program.

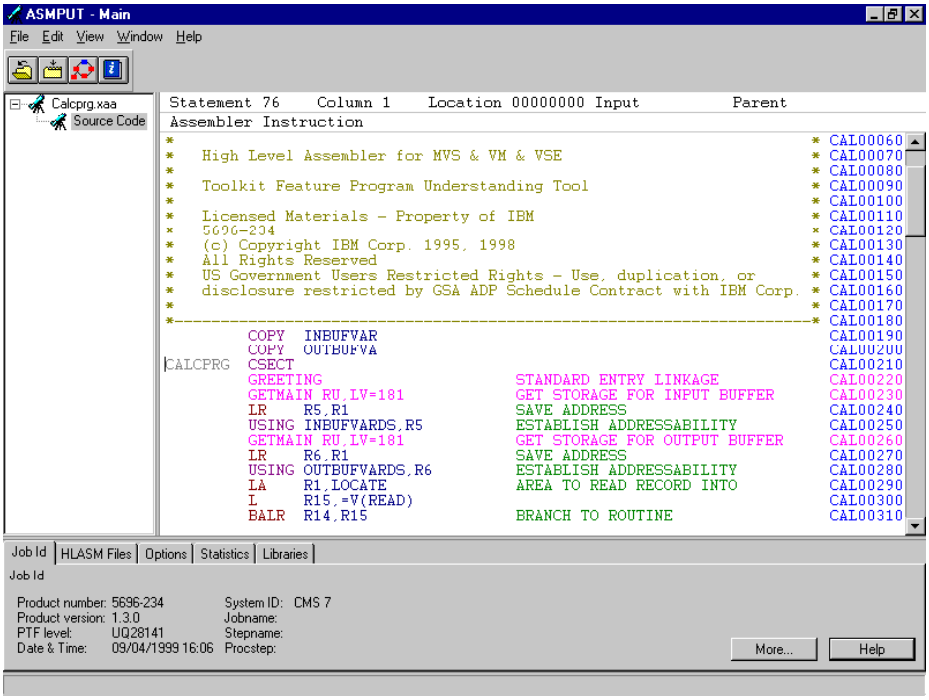
ASMPUT provides a text and a graphical display of programs and modules. These displays are linked. Selecting an element in one display automatically highlights the corresponding element in the other display. This cross-linking helps you quickly see the association between the assembled source code and the graphical control-flow representations of the program.

At any time, you can narrow or expand the focus of your analysis by zooming in on or out from areas of particular interest in the control flow graph.

ASMPUT comes with online help, sample files, and a slide-show demonstration to quickly familiarize you with the techniques required to use ASMPUT.

Installation is simplified by packaging all Toolkit components as host files; ASMPUT files are then downloaded to the workstation.

ASMPUT Main Window



HLASM Toolkit TKIT-5

The ASMPUT Main Window

This is the initial window displayed when ASMPUT starts. You control the session from this window. When this window closes, the complete ASMPUT session is closed.

The default display shows all three window areas:

File list area

Shows the open SYSADATA (.XAA) files, and provides the means to remove (close) individual files.

Source display area

Shows the current source code and has the usual scroll bars for moving through the source. Right clicking within this area displays a pop-up menu that allows for modification of the display content or searching the source for either a text string or an error message.

Change the display font, style and size by selecting the fonts option from the pulldown menu.

Information notebook

The notebook displays assembly time information. It is an optional window - it may be removed from the display by removing the check mark on the pulldown menu. The different tabs show different information:

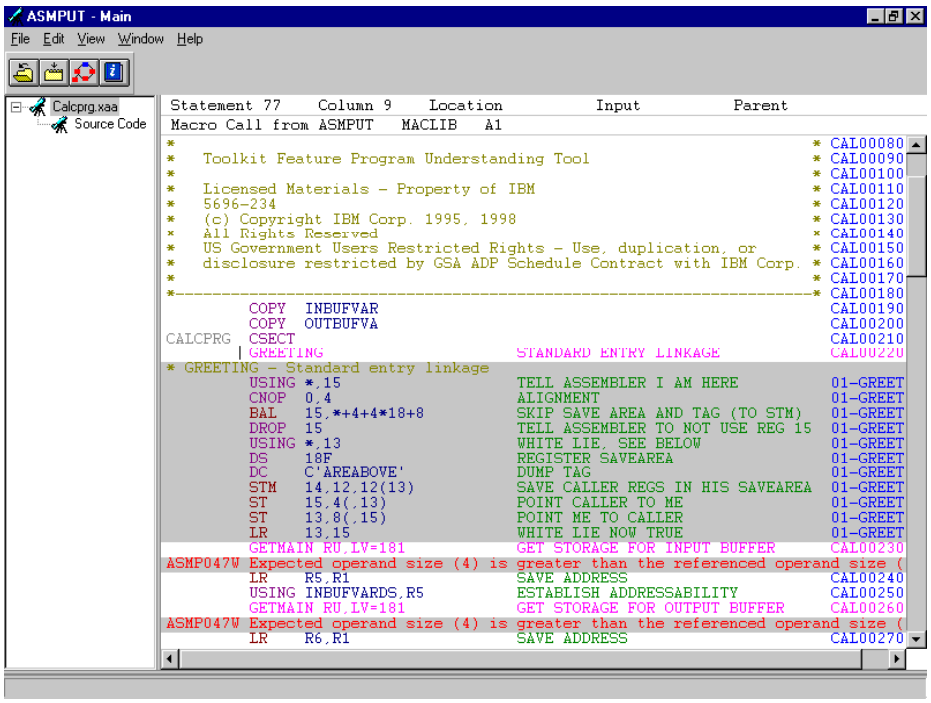
- Job ID

- I/O files
- Options
- Statistics
- Libraries

You can resize these areas by dragging their common borders.

You can open the online help from the menu, or call for help from information notebook displays.

Looking at Text Source



HLASM Toolkit TKIT-6

Looking at Text Source

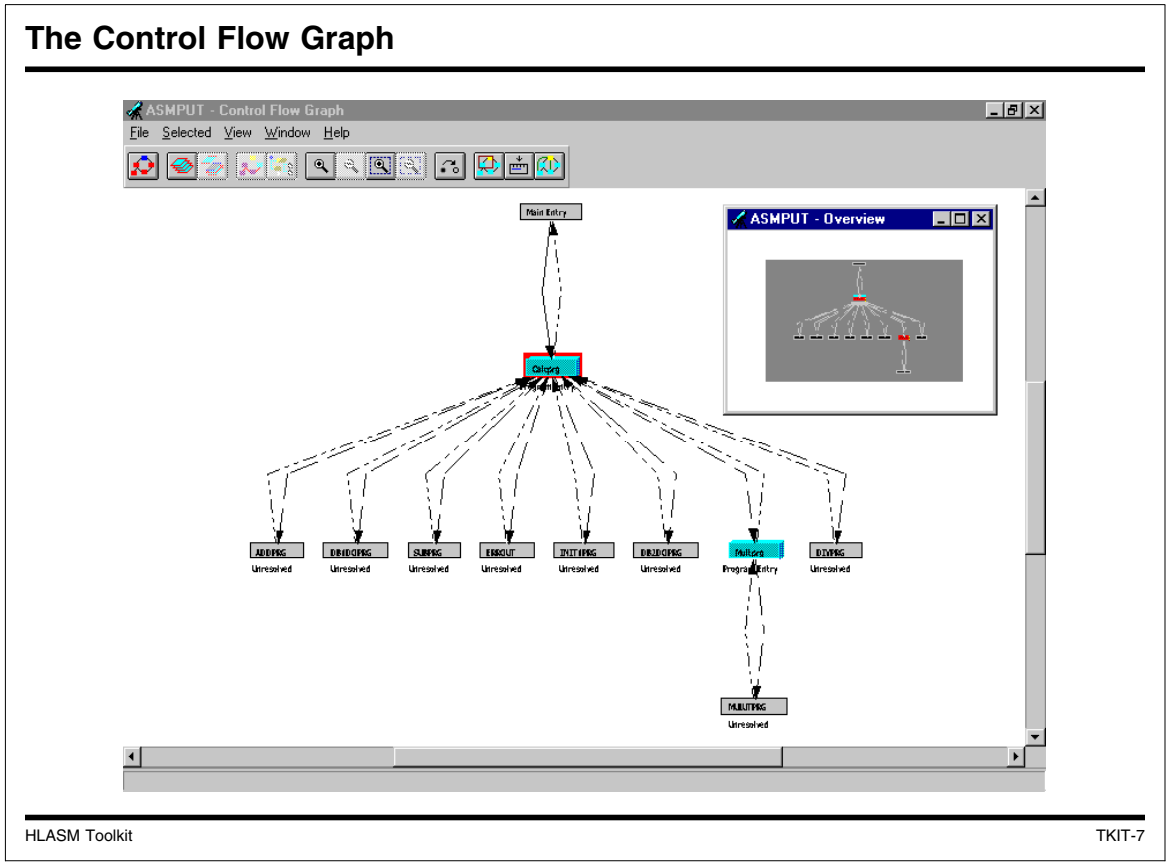
The **source code area** displays the source code listing of one program or module.

Lines of source code are color coded, to indicate their function.

Assembly diagnostics and analysis messages are displayed in red. You can toggle their display.

Macro calls and COPY segments are displayed in magenta (pink). You can display or hide all the lines in these calls or segments, by double-clicking the lines.

You can also invoke the **Find** function to look for an item of text in the currently-displayed lines.



The Control Flow Graph

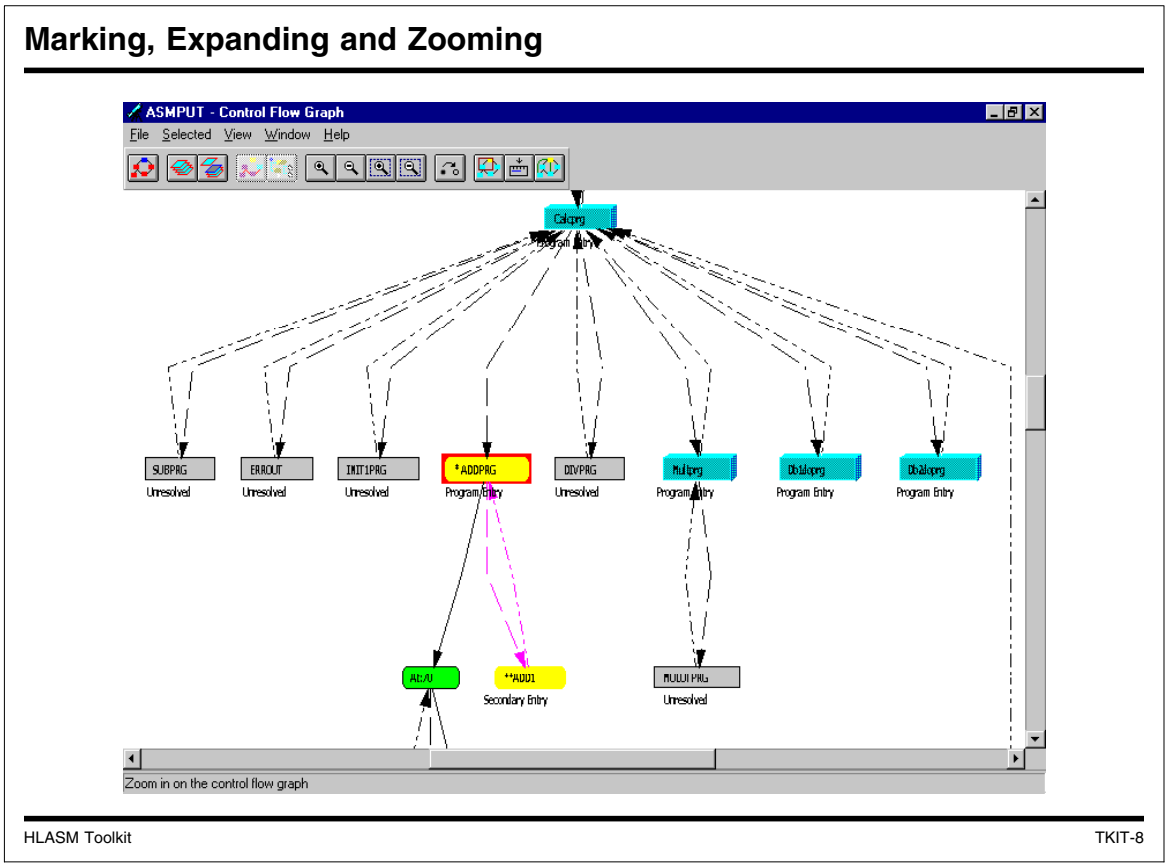
The control flow graph reveals the flow both within a module and within a set of modules.

There are many facilities to help navigate through the graph, including marking, expanding and collapsing nodes, zooming in and out, and removing or adding context. All of these facilities are available from the menu bar or from pop-up menus. The **overview window** shows the complete control flow graph and provides another way of controlling information displayed on the control flow graph.

The control flow graph is a set of nodes and arcs. Each node represents a segment of code. A node that is displayed as though it is in 3D can be “expanded,” to display a connected group of nodes and arcs. A node that is displayed as though it is in 2D cannot be expanded any further.

You can expand either a single (3D) node, or you can expand the entire control flow graph, either one layer, or to the maximum expansion. As you expand, more nodes are displayed.

To look more closely at the control flow graph, you zoom in. This doesn't change the structure of the graph at all, it just changes the size of the nodes and the font.



Marking, Expanding and Zooming

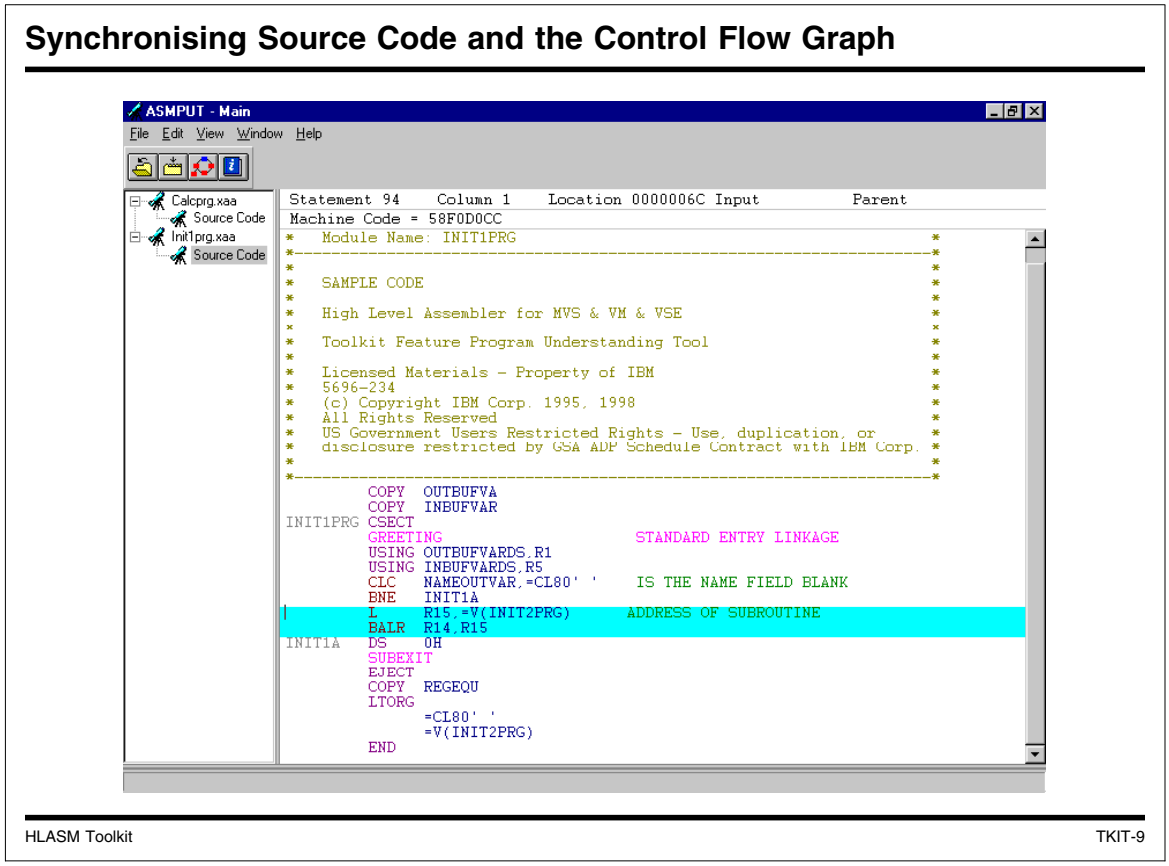
This slide shows what happens when a node is marked and expanded.

Marking a node changes its color to yellow. The color is retained when the node is expanded, which means that all of the resulting nodes (in this case, three) are colored yellow.

When you expand a node, you add more nodes to the overall control flow graph, thus making the graph more complex. One way to hide the complexity is to zoom in, so that most of the graph is not displayed because it “spills off” the edge of the window.

You can select one node by clicking it. The selected node is outlined in red.

Zooming in leaves the structure of the control flow graph unchanged, but magnifies the nodes and arcs.

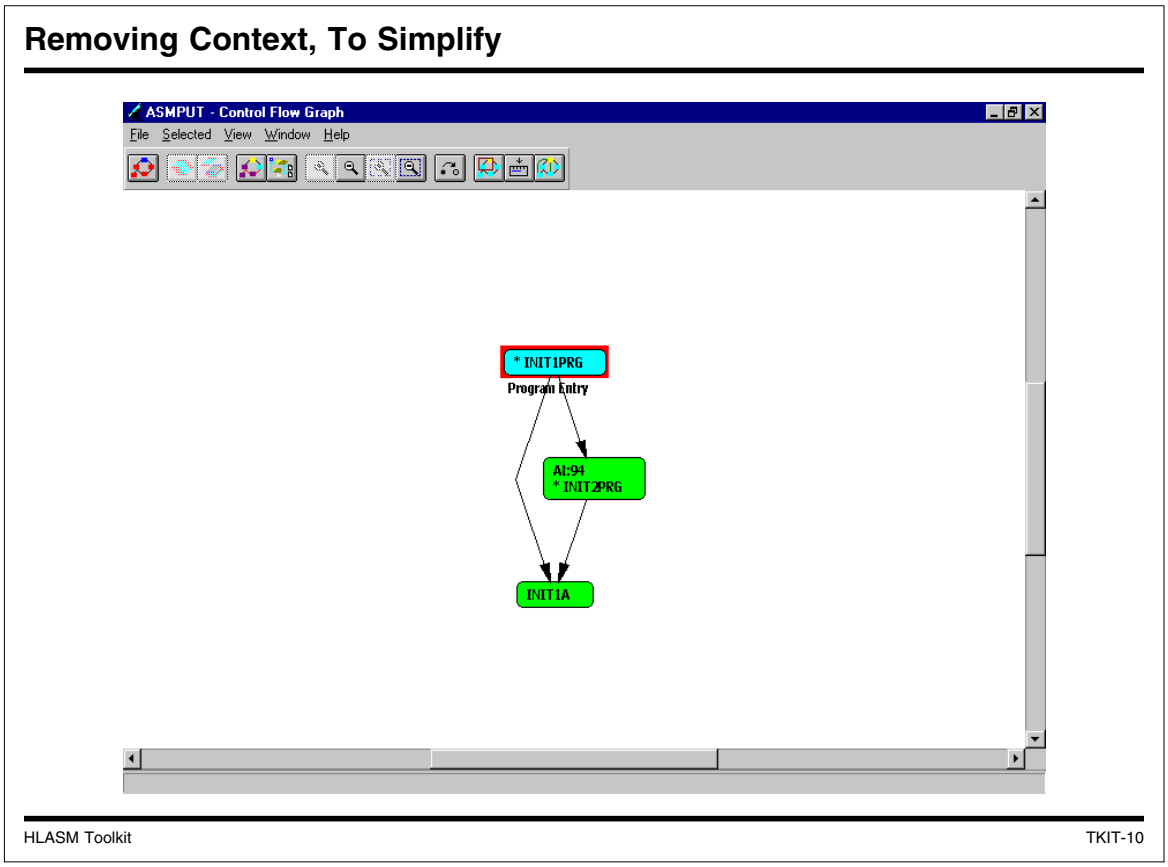


Synchronising Source Code and the Control Flow Graph

This slide shows the source code for the “Init1prg.xaa” program.

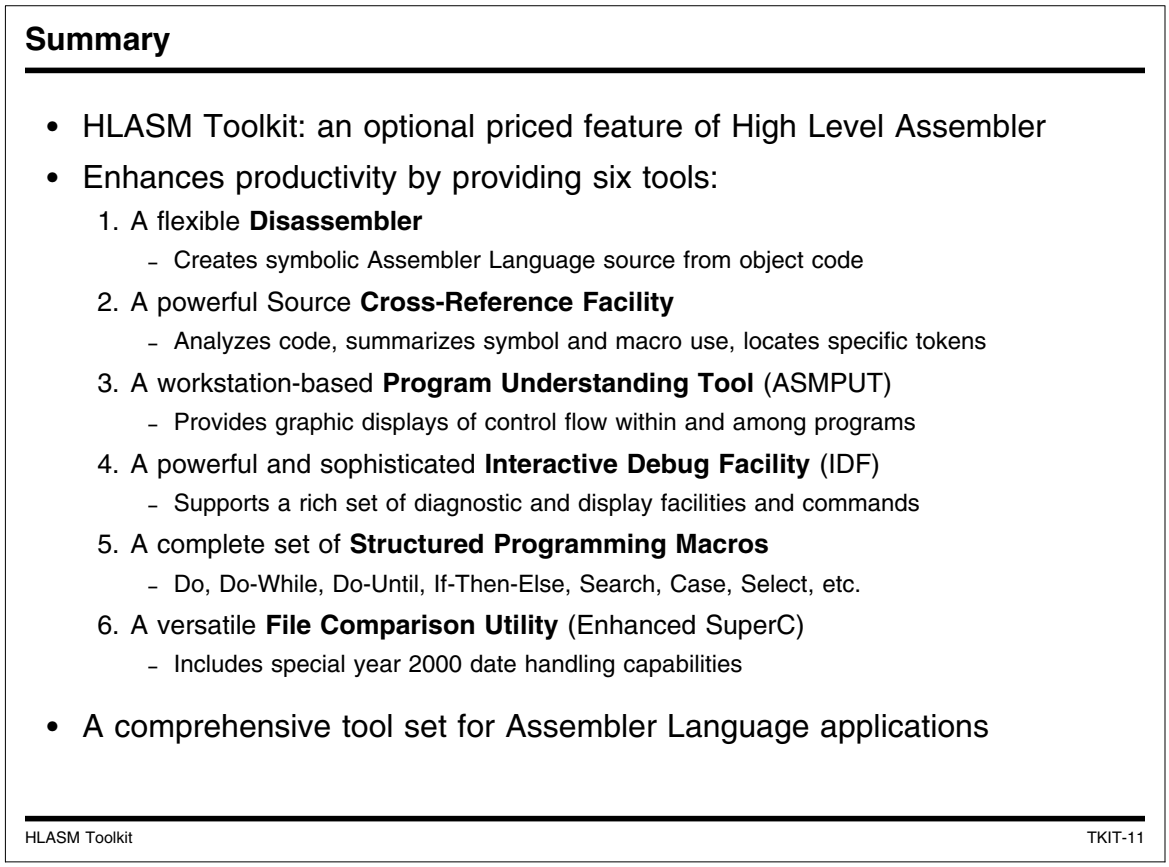
Two lines are highlighted in light blue. These are the lines that correspond to the selected node. (If you look at the top of the source code area, you can see `Statement 94`, which corresponds to the node name `At:94`.)

The link also works the other way—click an executable line of code, and the node that corresponds to it in the control flow graph is made the current node, and given a red surround.



Removing Context, To Simplify

When you remove the context for a node, only the nodes that are directly linked to the selected node are displayed. This simplifies the display of the control flow graph, and lets you focus on just the critical nodes. You may even end up with something as simple as this three-node graph!



Summary

The High Level Assembler Toolkit Feature is an optional, separately priced feature of IBM High Level Assembler (HLASM). It provides a powerful and flexible set of six tools to improve application recovery and development on OS/390, MVS/ESA, VM/ESA, and VSE/ESA systems. These productivity-enhancing tools are:

- **Disassembler**, a tool which converts binary machine language to Assembler Language source statements. It helps you understand programs in executable or object format, and enables recovery of lost source code.
- **Cross-Reference Facility**, a flexible source-code analysis and cross-referencing tool. It helps you determine variable and macro usage, analyze high-level control flows, and locates specific uses of arbitrary strings of characters.
- **ASMPUT**, a workstation-based program analysis tool. It provides multiple and “variable-magnification” views of control flows within single programs or across entire application modules.
- **Interactive Debug Facility**, a powerful and sophisticated symbolic debugger for applications written in Assembler Language and other compiled languages. It simplifies and speeds the development of correct and reliable applications. (It is not intended for debugging privileged or supervisor-state code.)
- **Structured Programming Macros**, a complete set of macro instructions that implement the most widely used structured-programming constructs (IF, DO,

CASE, SEARCH, SELECT). These macros simplify coding and help eliminate errors in writing branch instructions.

- **File Comparison Utility** (known as "Enhanced SuperC"), a versatile file searching and comparison tool. It can scan or compare single file or groups of files with an extensive set of selection and rejection criteria. Typical uses include comparing an original source file with a modified source file, or a pre-migration application output file with a post-migration output file. Newly added functions include "smart comparisons" of date fields to assist year 2000 migrations.

Together, these tools provide a powerful set of capabilities to speed application development, diagnosis, and recovery.

Appendix: Usage Scenarios

This appendix describes how you might use the High Level Assembler Toolkit Feature for typical program recovery, development, analysis, conversion, and maintenance tasks. These three scenarios are especially relevant to solving the problems of migrating Assembler Language applications to support the year 2000 and beyond, and show how the challenges of year 2000 tasks can be completed with greater speed and simplicity using the Toolkit Feature.

The Toolkit Feature components are described in three scenarios:

- Recovery and reconstruction of symbolic Assembler Language source code.
- Analysis and understanding of complex Assembler Language programs.
- Modification, testing, and validation of applications.

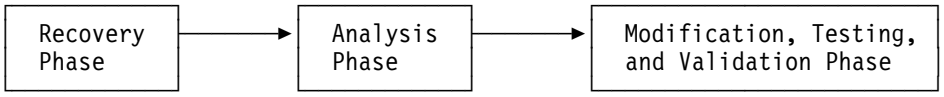


Figure 1. Typical Scenarios for Toolkit Feature Usage

1. ***Recovery and reconstruction*** of Assembler Language source statements from object/load modules for which the original source is lost. The Disassembler initially produces non-symbolic Assembler Language source from object code. You can add control statements iteratively to help define code, data, USINGs, labels, and DSECTs symbolically.
2. ***Analysis and understanding*** of Assembler Language source programs can benefit from three Toolkit components: the Cross-Reference Facility, ASMPUT, and the Interactive Debug Facility.
   a. The Cross-Reference Facility source analyzer and token scanner can be used to locate important symbols, user-selected tokens, macro calls, inter-module references, and other helpful data. ASMXREF can also create an "impact-analysis" file for input to a spreadsheet application for effort estimation and impact assessment. Another ASMXREF output is a tagged Assembler Language source file: when assembled with the ADATA option, this file produces a SYSADATA file for you to use with ASMPUT.
   b. ASMPUT provides graphic displays of program structure, control flow, a simplified listing, and other views with any desired level of detail. With the

ADATA file created from the tagged source produced by ASMXREF, key areas of the program can be rapidly located and analyzed.

c. The Interactive Debug Facility is by design a “program understanding” tool that lets you monitor the behavior of programs at every level of detail. Data flows may be monitored and traced among registers and storage, even showing the operations of individual instructions!

Note that the combination of Disassembler, Cross-Reference Facility, and ASMPUT can be used to help reconstruct lost source in compiled High Level Languages.

3. ***Modification and testing*** of updated programs is simplified by using the powerful Interactive Debug Facility. At the same time, program logic can be simplified by replacing complex test/branch logic with the Structured Programming Macros.

   Validation: At each stage where the application has been changed, you will probably want to compare its “pre-modification” output to its “post-modification” output, retaining the output files (sometimes called “base logs”) for subsequent validation tests. The File Comparison Utility Enhanced SuperC is designed specifically for such tasks.

Recovery and Reconstruction

During the Recovery and Reconstruction phase, you typically begin with a program in object or executable format (except CMS MODULEs). Using the Disassembler and by providing suitable control statements, you can create an Assembler Language source program with as much structure and symbolic labeling as you like.

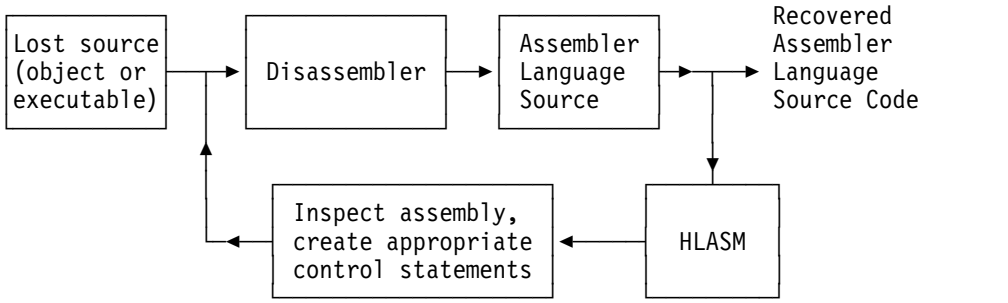


Figure 2. Toolkit Feature: Recovery and Reconstruction Scenario

The disassembly/analysis/description/assembly cycle may be repeated until satisfactory Assembler Language source code is obtained.

The initial steps do not require reassembly of the generated Assembler Language source, as appropriate control statements are usually easy to determine from the Disassembler's listing. As the recovered program approaches its final form, you should assemble it with HLASM to ensure the validity of your new source program.

Analysis and Understanding

The most complex aspect of application maintenance and migration is analyzing and understanding the code. There are three components of Toolkit Feature that can help:

- ASMXREF can locate all uses of a variable name or any character string. A tagged Assembler Language source program may also be produced.

- ASMPUT provides graphical views of control flows within and among programs and modules.
- The Interactive Debug Facility helps you monitor and track the behavior of individual instructions and data items.

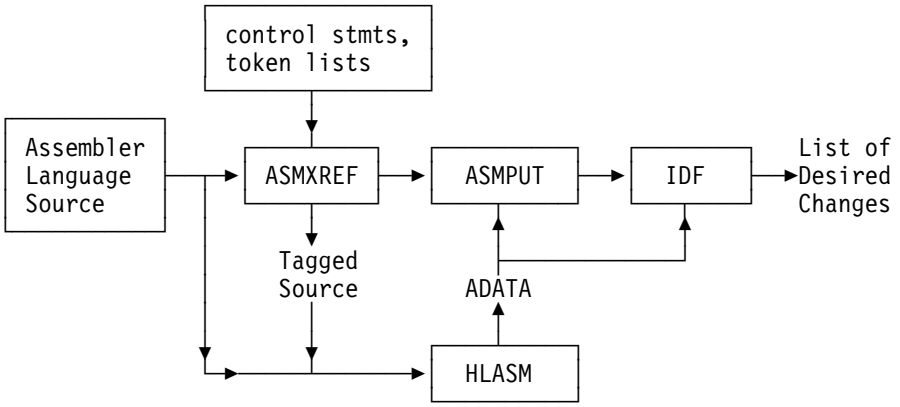


Figure 3. Toolkit Feature: Analysis and Understanding Scenario

While each of these components has valuable capabilities, using them in combination can provide great synergy in analyzing and understanding program behavior.

Modification and Testing

After you have used the Toolkit's disassembler, ASMXREF, and ASMPUT components to determine the needed modifications, the Structured Programming Macros can be added to simplify the coding and logic of the program.

The Enhanced SuperC comparison utility can then be used to compare the original and updated source files to validate the placement and coverage extent of all modifications.

You can then test the updated code using the rich and flexible features of the Interactive Debug Facility. After each assembly/debug cycle, you can further modify the source code, repeating the process until the completed application is accepted for installation in a production library.

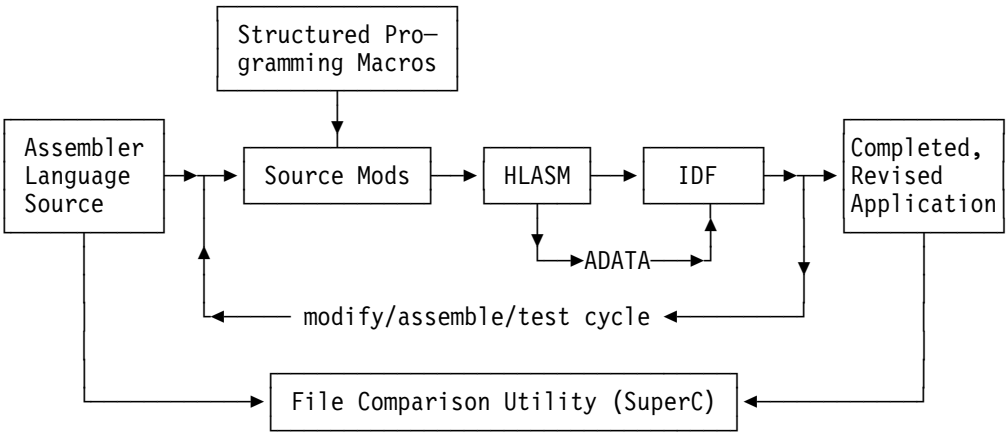


Figure 4. Toolkit Feature: Modification and Testing Scenario

Validation

After some set of modifications has been made to the application, you will probably need to validate its operation. Typical steps in such a process include the following:

1. Run the original unmodified program with a representative set of “old data,” and with the current date set to some manageable “current” date prior to the year 2000.
2. Run the modified program with the same set of “old data” and the same current date. (There are many techniques available for setting chosen “current” dates on a system.)
3. Use Enhanced SuperC to compare the outputs to ensure that no regressions have been introduced into the existing function of the application. If some date fields have been expanded (such as in report headings), use the date-format facilities of Enhanced SuperC to specify how they should be expanded and compared.
4. Run the modified program with the same “old data” and a new current date, set past 2000.
5. Use Enhanced SuperC to compare these new outputs with the previous two, using Enhanced SuperC's “aging” facilities to ensure correct current-date-dependent behavior.
6. Run the modified program with “new data” and the same new current date.
7. Use Enhanced SuperC to compare the outputs, using “aging” facilities to validate data-dependent behavior.

While not a complete migration plan, the above steps are typical of year 2000 migration activities. At each stage, the File Comparison Utility can provide powerful insights into the extent and correctness of code modifications.

Scenario Summary

These scenarios illustrate how the HLASM Toolkit Feature provides a varied and powerful set of tools supporting all aspects of application development, maintenance, enhancement, and testing. The following figure summarizes these capabilities:

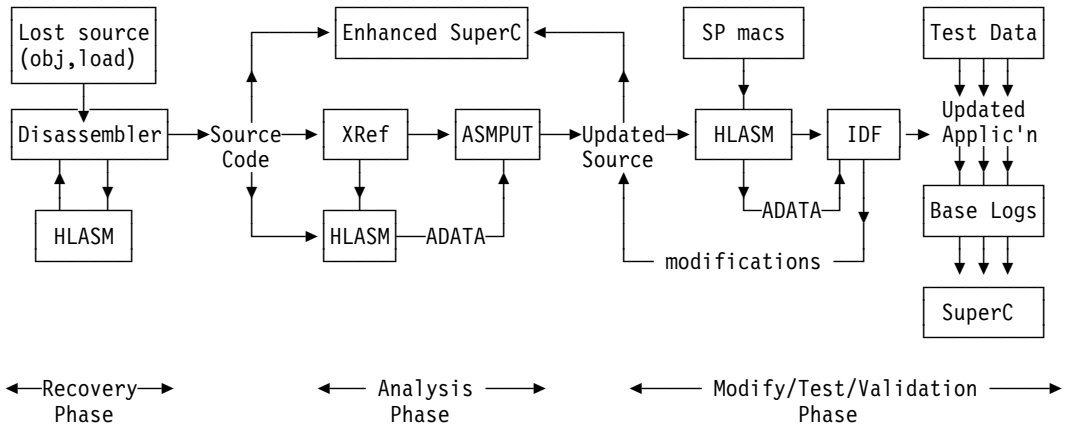


Figure 5. Toolkit Feature: Summary of Usage Scenarios