**ASSIST/UNA: A Windows-Based Emulator**

**of the ASSIST/I Assembler for the IBM/360**

Software Requirements Specification

Version 1.0.1

February 6, 2014

Travis Hunt, *Team Leader*

Michael Beaver, *Technical Writer*

Andrew Hamilton, *Software Quality Assurance*

Drew Aaron

Clay Boren

Chad Farley

University of North Alabama

CS 455: Software Engineering

Dr. Patricia L. Roden

Spring 2014

# Version History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version Number** | **Author(s)** | **Comments** |
| 02/05/2014 | Version 1.0 | Michael Beaver | Initial draft |
| 02/06/2014 | Version 1.0.1 | Michael Beaver | Added definitions, acronyms, and abbreviations;  Added project scope;  Added references;  Added user characteristics;  Added change management process; Added assembly instructions appendix;  Removed unnecessary sections |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

Version History i

1. Introduction 1

1.1 Purpose 1

1.2 Scope 1

1.3 Definitions, Acronyms, and Abbreviations 1

1.4 References 2

1.5 Overview 3

2. General Description 4

2.1 Product Perspective 4

2.2 Product Functions 4

2.3 User Characteristics 4

2.4 General Constraints 4

2.5 Assumptions and Dependencies 4

3. Specific Requirements 5

3.1 External Interface Requirements 5

3.1.1 User Interfaces 5

3.1.2 Hardware Interfaces 5

3.1.3 Software Interfaces 5

3.1.4 Communications Interfaces 5

3.2 Functional Requirements 5

3.2.1 <Functional Requirement or Feature #1> 5

3.2.2 <Functional Requirement or Feature #2> 6

3.3 Use Cases 6

3.3.1 Use Case #1 6

3.3.2 Use Case #2 6

3.4 Classes / Objects 6

3.4.1 <Class / Object #1> 6

3.4.2 <Class / Object #2> 6

3.5 Non-Functional Requirements 6

3.5.1 Performance 6

3.5.2 Reliability 6

3.5.3 Availability 6

3.5.4 Security 6

3.5.5 Maintainability 6

3.5.6 Portability 6

3.6 Design Constraints 7

3.7 Other Requirements 7

4. Deliverables 8

5. Change Management Process 9

5.1 Email Report Guidelines 9

6. Client-Developer Contractual Agreement 10

A. Appendices 11

A.1 Assembly Instructions to Implement 11

A.2 Diagrams 12

A.2.1 Frontend Diagram 12

A.2.2 Backend Diagram 12

A.3 Prototype Graphical User Interface Screen Captures 13

A.4 Features for Future Consideration 13

Index 14

# 1. Introduction

The introduction to the Software Requirement Specification (SRS) document should provide an overview of the complete SRS document. While writing this document please remember that this document should contain all of the information needed by a software engineer to adequately design and implement the software product described by the requirements listed in this document. (Note: the following subsection annotates are largely taken from the IEEE Guide to SRS).

## 1.1 Purpose

*What is the purpose of this SRS and the (intended) audience for which it is written.*

## 1.2 Scope

The ASSIST/UNA software is a Windows-based emulator of the ASSIST/I assembly language that will enable students to write assembly programs in a graphical environment. The ASSIST/UNA software will provide students with a subset of the functionality of the ASSIST/I assembler (see Sections 2 and 3). At this time, the ASSIST/UNA software is not meant to *fully* emulate all of the features of the ASSIST/I assembler; however, these features may be implemented later (see Appendix A.4).

The ASSIST/UNA software will be used by students and instructors in future CS 310 courses. This port to the Windows environment eliminates the need to use other emulating software, such as DOSBox, to write assembly programs. Also, the ASSIST/UNA IDE is designed to be user-friendly and to serve as a collection of useful tools, such as printing program reports. The ASSIST/UNA software will enable students to interact with an emulated ASSIST/I assembler in an intuitive and user-friendly environment.

## 1.3 Definitions, Acronyms, and Abbreviations

This section provides definitions of all terms, acronyms, and abbreviations required to properly interpret this software requirements specification document.

**1.3.1** **Assemble:** To translate source statements into corresponding machine code in the form of an object code program.

**1.3.2 Assembler:** A piece of software that assembles assembly source statements.

**1.3.3 ASSIST:** The Assembler System for Student Instruction and Systems Teaching for the IBM/360 (and IBM/370) developed by John Mashey and his students at Pennsylvania State University in the 1970s.

**1.3.4 ASSIST/I:** The version of ASSIST for personal computers.

**1.3.5 ASSIST/UNA:** The University of North Alabama emulator of the ASSIST/I assembler.

**1.3.6** **Client:** The person that has instigated development and has decided the project requirements.

**1.3.7 CS 310:** The Computer Science 310: Computer Organization & Assembly Language Programming course taught at the University of North Alabama.

**1.3.8 Developer:** The person, or persons, who actively design, implement, and maintain the ASSIST/UNA software and the accompanying documentation.

**1.3.9 DOSBox:** A Windows program that emulates the Windows DOS environment.

**1.3.10 Emulator:** Software on a hardware system that imitates the functionality of another software on its respective hardware system.

**1.3.11 GitHub:** A repository service (www.github.com) used to store remote copies of project source code and documentation.

**1.3.12 GUI:** Graphical User Interface.

**1.3.13 IDE:** Integrated Development Environment.

**1.3.14 UNA:** The University of North Alabama in Florence, Alabama.

**1.3.15 User:** The person, or persons, who interact directly with the ASSIST/UNA software.

**1.3.16 Team Foundation Server:** A Microsoft Visual Studio version-control system used to maintain and ensure the integrity of project source code.

**1.3.17 TFS:** Team Foundation Server.

**1.3.18 Windows:** The Microsoft Windows operating system environment.

## 1.4 References

This software requirements specification document shall be used in conjunction with the following publications and client handouts.

Client, Assembly Instructions to Implement.[[1]](#footnote-1)

Client, Initial Project Description.[[2]](#footnote-2)

Client, More Detailed Description.[[3]](#footnote-3)

IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications.[[4]](#footnote-4)

John R. Mashey and Graham Campbell, ASSIST Introductory Assembler User’s Manual.[[5]](#footnote-5)

J. R. Mashey, G. M. Campbell, and C. Forney, Jr., Assist: A Self Modifiable Assembler for Instructional Purposes.[[6]](#footnote-6)

Jay Moseley, ASSIST – Assembler System for Student Instruction & Systems Teaching.[[7]](#footnote-7)

## 1.5 Overview

*This subsection should:*

*(1) Describe what the rest of the SRS contains*

*(2) Explain how the SRS is organized.*

# 2. General Description

*This section of the SRS should describe the general factors that affect 'the product and its requirements. It should be made clear that this section does not state specific requirements; it only makes those requirements easier to understand.*

## 2.1 Product Perspective

*This subsection of the SRS puts the product into perspective with other related products or*

*projects. (See the IEEE Guide to SRS for more details).*

## 2.2 Product Functions

This subsection of the SRS should provide a summary of the functions that the software will perform.

## 2.3 User Characteristics

The ASSIST/UNA software is to be used by the students and instructor of the CS 310 course at UNA, or in similar courses at other universities. Student users should have a basic understanding of programming fundamentals and basic understanding of the ASSIST/I assembly language. Although not required, student users with experience with IDEs will be benefited. Student users lacking experience with IDEs will find the ASSIST/UNA IDE to be intuitive and user-friendly. Instructor users should have a strong understanding of the ASSIST/I assembly language. The ASSIST/UNA IDE will enable instructor users to easily open, examine, and test student users’ programs. Instructor users will also be able to write their own programs.

## 2.4 General Constraints

*This subsection of the SRS should provide a general description of any other items that will*

*limit the developer’s options for designing the system. (See the IEEE Guide to SRS for a partial list of possible general constraints).*

## 2.5 Assumptions and Dependencies

This subsection of the SRS should list each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the SRS. For example, an assumption might be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS would then have to change accordingly.

# 3. Specific Requirements

This will be the largest and most important section of the SRS. The customer requirements will be embodied within Section 2, but this section will give the D-requirements that are used to guide the project’s software design, implementation, and testing.

Each requirement in this section should be:

* Correct
* Traceable (both forward and backward to prior/future artifacts)
* Unambiguous
* Verifiable (i.e., testable)
* Prioritized (with respect to importance and/or stability)
* Complete
* Consistent
* Uniquely identifiable (usually via numbering like 3.4.5.6)

Attention should be paid to the carefuly organize the requirements presented in this section so that they may easily accessed and understood. Furthermore, this SRS is not the software design document, therefore one should avoid the tendency to over-constrain (and therefore design) the software project within this SRS.

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

### 3.1.2 Hardware Interfaces

### 3.1.3 Software Interfaces

### 3.1.4 Communications Interfaces

## 3.2 Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

### 3.2.1 <Functional Requirement or Feature #1>

3.2.1.1 Introduction

3.2.1.2 Inputs

3.2.1.3 Processing

3.2.1.4 Outputs

3.2.1.5 Error Handling

### 3.2.2 <Functional Requirement or Feature #2>

…

## 3.3 Use Cases

### 3.3.1 Use Case #1

### 3.3.2 Use Case #2

…

## 3.4 Classes / Objects

### 3.4.1 <Class / Object #1>

3.4.1.1 Attributes

3.4.1.2 Functions

<Reference to functional requirements and/or use cases>

### 3.4.2 <Class / Object #2>

…

## 3.5 Non-Functional Requirements

Non-functional requirements may exist for the following attributes. Often these requirements must be achieved at a system-wide level rather than at a unit level. State the requirements in the following sections in measurable terms (e.g., 95% of transaction shall be processed in less than a second, system downtime may not exceed 1 minute per day, > 30 day MTBF value, etc).

### 3.5.1 Performance

### 3.5.2 Reliability

### 3.5.3 Availability

### 3.5.4 Security

### 3.5.5 Maintainability

### 3.5.6 Portability

## 3.6 Design Constraints

Specify design constrains imposed by other standards, company policies, hardware limitation, etc. that will impact this software project.

## 3.7 Other Requirements

Catchall section for any additional requirements.

# 4. Deliverables

aaa

# 5. Change Management Process

Before reporting a change or an error, developers must ensure they are consulting the most recent version of the software requirements specification document. If a change is required or an error is encountered, this software requirements specification document will be updated by following this update process. Developers shall notify the technical writer and the team leader via email. The email will detail the section(s) and line number(s) that need to be changed. In addition, the email will contain detailed descriptions of each change or error. The reporting developer shall also provide a detailed suggested correction. The technical writer will review the correction reports, approve the corrections, and update this document as necessary. The new version of this document will be uploaded to the GitHub server, and each developer will be notified via email.

## 5.1 Email Report Guidelines

If a developer needs to report changes or errors in the software requirements specification document, he shall follow these guidelines:

* The email will be sent to the technical writer and the team leader.
* The email subject will read: “CS 455 – Spec Doc Changes.”
* The email will specify the version of the software requirements specification document.
* The email will contain the following sections for each change or error:
  + Section Number
  + Line Number
  + Detailed description of the change or error
  + Detailed suggested correction
* Developers should send themselves a carbon copy for their own records.

Each section of the email should be appropriately labeled. For example, the “Section Number(s)” section should be labeled “**Section Number(s)**.” Note that each reported change or error will need its own set of the aforementioned sections (with appropriate labels).

# 6. Client-Developer Contractual Agreement

The client (Dr. Patricia L. Roden) is satisfied that this software requirements specification document meets the client’s requirements and needs as specified at this time. The developers (listed below) recognize the client’s right to request modifications of the project requirements. The client will ensure the modifications are necessary and will notify the developers at least one week in advance. The client hereby approves this software requirements specification document in its current form. The developers agree thusly to deliver the agreed upon deliverables on or before April 29, 2014.

The client agrees to fairly assess the work delivered by the developers, in accordance with the specified requirements within this document. Additionally, the client agrees to assign to each developer the appropriate grade mark for his individual and overall contributions to the project. While grading the developers’ work, the client agrees to consult each developer’s evaluations. The signatures below hereby bind the client and the developers to this contractual agreement.

*Dr. Patricia L. Roden, Client* Date

*Travis Hunt, Team Leader* Date

*Michael Beaver, Technical Writer* Date

*Andrew Hamilton, Software Quality Assurance*  Date

*Drew Aaron* Date

*Clay Boren* Date

*Chad Farley* Date

# A. Appendices

Appendices may be used to provide additional (and hopefully helpful) information. If present, the SRS should explicitly state whether the information contained within an appendix is to be considered as a part of the SRS’s overall set of requirements.

*Example Appendices could include (initial) conceptual documents for the software project, marketing materials, minutes of meetings with the customer(s), etc.*

## A.1 Assembly Instructions to Implement

This appendix contains the list of required ASSIST/I assembly instructions to be implemented in the ASSIST/UNA emulator. Table A.1.1 lists the types of instructions supported by ASSIST/UNA. Table A.1.2 lists each ASSIST/I instruction’s mnemonic, description, instruction type, and basic form.

Table A.1.1: ASSIST/I instruction types supported by ASSIST/UNA.

|  |  |
| --- | --- |
| **Instruction Type** | **Type Meaning** |
| RR | Register-Register. Values are taken from registers, manipulated, and the result is stored into a register (e.g., AR 3,4). |
| RS |  |
| RX |  |
| SS |  |
| X\* | An instruction whose mnemonic is preceded by an “X” is a special macro instruction that actually comprises several instructions (e.g., XDECO). |
|  |  |
|  |  |

Table A.1.2: ASSIST/I Instructions to be implemented.

|  |  |  |  |
| --- | --- | --- | --- |
| **Mnemonic** | **Description** | **Type** | **Basic Form** |
| A | Add |  |  |
| AP | Add Packed |  |  |
| AR | Add Register | RR |  |
| BAL | Branch And Link |  |  |
| BALR | Branch and Link Register |  |  |
| BC | Branch on Condition |  |  |
| BCR | Branch on Condition Register |  |  |
| BCT | Branch on Count |  |  |
| BCTR | Branch on Count Register |  |  |
| BXH | Branch Higher |  |  |
| BXLE | Branch Less than or Equal to |  |  |
| C |  |  |  |
| CLC |  |  |  |
| CLI |  |  |  |
| CP | Compare Packed |  |  |
| CR | Compare Register |  |  |
| D | Divide |  |  |
| DP | Divide Packed |  |  |
| DR | Divide Register |  |  |
| ED | Edit |  |  |
| EDMK | Edit and Mark |  |  |
| L | Load |  |  |
| LA | Load Address |  |  |
| LM | Load Multiple |  |  |
| LR | Load Register |  |  |
| M | Multiply |  |  |
| MP | Multiply Packed |  |  |
| MR | Multiply Register |  |  |
| MVC |  |  |  |
| MVI |  |  |  |
| N |  |  |  |
| NR |  |  |  |
| O |  |  |  |
| OR |  |  |  |
| PACK | Pack |  |  |
| S | Subtract |  |  |
| SP | Subtract Packed |  |  |
| SR | Subtract Register |  |  |
| ST | Store |  |  |
| STM | Store Multiple |  |  |
| UNPK | Unpack |  |  |
| XDECI | Convert Input to Decimal |  |  |
| XDECO | Convert Output to Decimal |  |  |
| XDUMP | Dump Memory and Registers |  |  |
| XPRNT | Print |  |  |
| XREAD | Read Input |  |  |
| ZAP | Zero, Add Packed |  |  |

## A.2 Diagrams

### A.2.1 Frontend Diagram

### A.2.2 Backend Diagram

## A.3 Prototype Graphical User Interface Screen Captures

## A.4 Features for Future Consideration

# Index

1. Available on ANGEL. See Appendix A.1. [↑](#footnote-ref-1)
2. See footnote 1. See Sections 2 and 3. [↑](#footnote-ref-2)
3. See footnote 1. See Sections 2 and 3. [↑](#footnote-ref-3)
4. Available at http://standards.ieee.org/findstds/standard/830-1998.html. This IEEE guide is used throughout this software requirements specification document. [↑](#footnote-ref-4)
5. Available at http://www.cbttape.org/features/assistmn.htm. See Section 2. [↑](#footnote-ref-5)
6. Available at http://dl.acm.org/citation.cfm?id=569933. See Section 2. [↑](#footnote-ref-6)
7. Available at http://www.jaymoseley.com/hercules/compiling/compile.htm#topic16. See Section 2. [↑](#footnote-ref-7)