Class Notes for Assembly Language Programming on the Compact Computer 40

Consumer Products Division

June 18,1984

Revision 1.10

Course Overview

The course is being held to teach TMS7000 assembly language within the hardware and software environment of the Compact Computer 40. The student will learn the fundamentals of assembly language, learn how to design, program, and debug programs, and learn how to choose between BASIC, assembly language or a combination of the two for a particular application. The course will assume a working knowledge of BASIC programming.

The course will consist of a number of programming exercises and four major projects. The projects are:

1. Convert a simple BASIC language benchmark program to assembly lanuguage to learn the fundamenatals of assembly language and understand the speed differences between the two.

2. Convert a more complex mathematical benchmark to assembly language to broaden the understanding of speed tradeoffs between BASIC and assembly language.

3. Write a simple memory test in assembly language to illustrate an application where assembly language is distinctly superior to BASIC.

4. Write an application which does text compression using a variable length coding scheme. This illustrates the advantage of assembly language where the application requires bit manipulation and would be very difficult in BASIC.

Ideas from the class for alternative or additional projects are welcomed.

There will be homework assignments for each class including the first one. The homework will be informally turned in for review to see if additional class discussion is required and see if the students are keeping up. The class will progress rapidly and very little will be learned without completing the homework assignments.

I will be available to provide assistance with the homework as necessary.

SECTION 1

Assignment for June 18

1.1 Purpose

This assignment is designed to generate familiarity with the equipment. It also will serve to familiarize the student with programming BASIC on the CC-40. The assembly language assignment will not require knowledge of the language. The BASIC programming assignments will provide programs which will be used later.

1.2 Preparation

Connect the equipment together. FORMAT a diskette for saving the programs written in this course.

1.3 Programming

Enter the following BASIC programs into the CC-40. RUN them and record the time each took to run. These programs are the simplest benchmark programs. They are often used to determine the relative speed of a BASIC language even though they give a very poor indication of performance in real applications. Later we will compare the time these benchmarks took with equivalent programs in assembly language.

```
100 REM Benchmark Program 1
110 FOR I=1 TO 2000:NEXT I

100 REM Benchmark Program 2
110 FOR I=1 TO 200:PRINT I:NEXT I
```

Enter the following BASIC program into the CC-40. In the lines 120 and 130 the space before the minus sign should contain the exponentiation symbol ([SHIFT]6). The text processor used to produce this document will not print that character. RUN it and record the time it took to run. This program computes the present value of a bond with periodic coupons. This is a better benchmark then the previous programs because it is similar to the operations performed in many applications. This program will also be used later to examine speed improvement using assembly language. SAVE this program on the diskette for future use.

```
90  REM Benchmark Program 3
100 M=20000:N=50:I=1400:Y=.08:R=1+Y
110 SUM=0:FOR J=1 TO N:SUM=SUM+R -J:NEXT J
120 SUM=I*SUM+M*R -N
130 PRINT SUM:PAUSE
```

Enter the following program and record the time it takes to run. SAVE it on diskette also. This program is a very simple memory test.

```
90  REM Benchmark Program 4
100 FOR I=4000 TO 4999:CALL POKE(I,I):NEXT I
110 FOR I=4000 TO 4999:CALL PEEK(I,J):K=I-INT(I/256)*256
120 IF J=K THEN NEXT I ELSE PRINT "Test Failed at";I:PAUSE
130 PRINT "Test Complete":PAUSE
```

Read Chapter 1 of the Editor/Assembler User's Guide. Do the steps described to enter and run the "Beep" demonstration. Record any problems you had in following the example. Answer the following additional questions through examination of the example and experimentation. You will have to read more about the editor in Chapter 4 to make changes to the program.

1. What would you change to make the beep last a longer or shorter period of time? How much can you change this?

2. Can you make the pitch of the beep be lower or higher? How much?

3. This subprogram can be CALLed from a BASIC program. Can you write a BASIC program which spaces beeps to produce sounds which remind you of a song?

1.4 Items to Turn In

Answer the following questions:

1. How long did each BASIC program take to run?

2. How does the memory test program work?

3. Specifically what data values get stored in the following locations: 4000, 4200, 4300?

4. (Optional) Make a guess for each BASIC program what the execution would be if it were written in assembly language.

Turn in listings of the BASIC programs and at least one listing of the assembly language program. Turn in your answers to the BEEP program questions.

1.5 Class Discussion

I will produce notes for the first class which augment the information in the Reference Guide. There is no reading assignment to prepare for the discussion unless I get the notes out before the class. We will discuss:

1. Programs and subprograms.

2. CC-40 memory discussion.

3. Different places to run programs.

4. Program headers.

5. Writing subprograms and programs for internal memory and cartridges.

6. TMS7000 assembly language.

SECTION 2

Assignment for June 25

2.1 Purpose

This assignment will increase the student's confidence by starting to write some assembly language. There will also be considerable reading for this assignment.

2.2 Preparation

Read Appendices A and B of this document and the Reference Manual pages listed:

- Chapter 1
- Chapter 2 pages 1-14
- Chapter 3 pages 1-26
- Chapter 4 (skim for familiarity)
- Chapter 5 pages 9-18
- Chapter 6
- Chapter 7 (skim for familiarity)
- Appendix A
- Appendix F

Locate and review Figure 2-1 on the last page of the E/A User's Guide. This Op-code map is very useful to help in programming. Also read pages 1 through 16 of the "BASIC Program Image" handout.

2.3 Programming

Convert the BEEP subprogram into a main program to run from internal RAM. This will involve changes in the header and a change in the calling conventions. This is a Level Zero program (see page 3-9) and returns to the system with RETSYS (see page 4-39).

Write a program which does the same thing as benchmark 1. Try this three ways. First do it using the MOVD, DECD and JC instructions. Then do it using CLR, ADD, ADC, and CMP instructions. These use integer math while the BASIC benchmark uses floating point. For the third try use floating point. The routines CLRFAC, GCONA, FADD, and FLTCMP may be useful.

Make a variant of benchmark 1 which is a subprogram using floating point and gets the stopping loop value from an argument passed to the subprogram. The NUMHEX example on page 5-11 will give some code ideas for acquiring the argument. Don't forget to check for the right parenthesis. Pages 3-13 through 3-18 are relevant to this assignment. The statement level temporary area can be used to store the argument (see page 3-1).

2.4 Items to Turn In

Turn in listings for each of the programming tasks. Turn in execution times for the benchmarks.

Answer the following question:

1. What is the hexidecimal floating point representation for each of the following numbers: 3.1415923, -3.1415923, 4.7, 47, 470, .47, .047, 123456789, 1234567890.

2.5 Class Discussion

We will discuss the BASIC system environment and floating point mathematics.

SECTION 3

Assignment for July 2

3.1 Purpose

3.2 Preparation

3.3 Programming

3.4 Items to Turn In

Answer the following questions:

3.5 Class Discussion

We will discuss:

APPENDIX A

Documentation Conventions

A.1 Byte Addressing

Bytes contain 8 bits. The bits are numbered 0 through 7. The least significant (right hand) bit is number 0 and most significant (left hand) bit is number 7.

A.2 Notes on Memory Diagrams

Various figures in this document describe data structure in memory. These are drawn to a consistant set of standards which need to be understood.

1. All diagrams are drawn with the highest numbered address at the top of the diagram and the lowest numbered address at the bottom. It is important to remember that assembly language code is written from low address to high which is bottom to top in the diagrams.

2. Items which are called "addresses," "words," or "displacements" are each two byte long. Unless specified otherwise the the most significant byte of these two-byte values is at the lowest address (as would be generated with a DATA directive).

A.3 Terminology

* Cartridge - a module which plugs into a computer and can contain RAM, ROM, or EPROM.

* Internal memory - internal RAM.

* Internal RAM - the RAM memory built into the product as opposed to RAM memory which can be added externally.

* Internal ROM - the ROM memory built into the product which contains the system software.

APPENDIX B

Additional Notes on Program Headers

These notes supplement the information in Chapter One of the Reference Manual. They should preferably be read first.

B.1 Types of Programs

A program can be either a subprogram which is accessed with the CALL statement or a main program which is accessed with the RUN statement. Programs can reside in one of three locations in the CC-40:

* Internal memory in the main program area.

* Internal memory in the subprogram area.

* Cartridge memory.

B.1.1 Main Program Area.

Only one main program can be contained in internal RAM. When a main program is loaded any main program already in memory is overwritten. Main programs in either BASIC or assembly language are loaded with the OLD command. The header of a BASIC language main program to be run in internal memory is significantly different then the header of an assembly language program. The header of an assembly language program to be run from main memory does not need a program (in fact it is not used). However, the "Program Entry Address" pointer must be at the correct location.

B.1.2 Subprogram Area.

This area resides in the low memory addresses of internal RAM above the system reserved area. Only subprograms written in assembly language can be loaded into this area. They are loaded with the CALL LOAD statement. Unlike main programs several subprograms can be loaded into memory either together or separately.

Subprograms loaded into this area are not destroyed when a main program is load into internal RAM. Only a NEW ALL or other form of system initialization will remove the subprograms.

The CALL LOAD command will link the subprogram headers together to form a "linked list." If more than one subprogram is assembled from one source module the programmer must link the headers together within the assembly.

B.1.3 Cartridge Memory.

A cartridge can hold a number of BASIC and assembly language main programs or subprograms. Unlike internal memory, a cartridge can hold a BASIC subprogram separate from the main program which calls. Assembly language can be loaded into a cartridge with the utilities provided with Editor/Assembler. BASIC main programs and subprograms can be loaded with the utilities provided with the Cartridge Development System.