Priyankkumar Shah

Stevens Institute of Technology

**"Reading a Text File with 68K Assembly”**

**Compliled successfully without any error or warning**

**A screenshot of a cell phone

Description automatically generated**

**CODE**

\*----------------------------------------------------------

\* Title :

\* Written by :

\* Date :

\* Description:

\*-----------------------------------------------------------

ORG $1000

START: ; first instruction of program

\*Put program code here

lea filename , a1

move #51, d0

trap #15

move.l #filesize, d2

lea buffer, a1

move #53, d0

trap #15

move #5, d3 ; counter

convert\_loop:

tst.b d3

beq done

sub.b #$30, (a1)+

sub.b #1, d3

bra convert\_loop

done:

SIMHALT ; halt simulator

org $2000

\* Put variables and constants here

filename dc.b 'supriti.txt', 0

buffer ds.b 80

filesize dc.b 80

END START ; last line of source

**Successful output**

**A screenshot of a computer

Description automatically generated**

**Supplemental (course review) questions**

1. **Please explain how to transform an assembly language program into a binary executable file using basic reference tables.**

**Answer:-**

Assembly code is converted into executable machine code by a utility program  referred to as an  assembler. The conversion process is referred to as assembly or *assembling* the source code. Assembly language usually has one statement per machine instruction, but comments and statements that are assembler directives, macros and symbolic labels of program and memory locations are often also supported.

Each assembly language is specific to a particular computer architecture and sometimes to an operating system. However, some assembly languages do not provide specific syntax for operating system calls, and most assembly languages can be used universally with any operating system, as the language provides access to all the real capabilities of the processor, upon which all system call mechanisms ultimately rest. In contrast to assembly languages, most  high-level programming languages are generally portable across multiple architectures but require interrupting or compiling.

1. **Please explain the formats of the different types of data (image, video, audio and alphanumerical, integers, floating point numbers).**

**Answer:-**

Image data: two types, bitmap and object. Can be encoded using several of the standard formats, GIF, TIFF, PCX, Windows Bitmap, etc.

Video Data: sampled in real time, 10’s of times per second. They generate huge amounts of data and require high speed I/O devices. It is processed using 2 techniques, streaming or locally stored.

Audio data is sampled in real time at thousands of times per second. It is encoded using some formats including MOD, MIDI, VOC, WAV, MP3.

Alphanumeric data generates low volumes of data so it requires a low speed I/O device that only needs 1 or 2 bytes. It is encoded using some of the formats: ASCII, EBCDIC, or Unicode

**Source:-**

<https://en.wikipedia.org/wiki/Assembly_language>