Team: Norman Hansen, Konstantin Stekhov

Program Description

Before running the program a user needs to load the test data into the program so the decompiler knows what to decode.

The program starts out by displaying a message that describes the name of the project a little description about it. Then the program prompts a user for a starting address and validates each character of the address while converting it from ASCII to HEX. After converting the address to HEX without errors it checks if the length of the address is between 0 and 8 characters. If it is not then the program displays the appropriate error and prompts a user for a new starting address. Next, the program checks if the address is even or not. If it is odd the program displays the error and asks for a new starting address. Then the starting address is validated so that its HEX value is between 7000 and FFFFFE.

After the user gave a valid starting address the program prompts for the ending address. The ending address goes through the same validation routine as for the starting address. In addition to that it is also validated so that it is larger than the starting address. If any the user gave an invalid ending address the program would prompt for both new starting and ending addresses.

When both starting and ending addresses are entered the program saves them in the address registers and starts executing the data that was loaded into the program. First it increments the starting address by word and saves the whole instruction that the word holds in the data register. Then it gets the first 4 bits from the instruction and the DECODE_OPCODE subroutine matches the data with instructions that have the same beginning part. Once it is matched then other functions will look at the next part part (depending on the instruction) of the instruction so that the disassembler knows what instruction is being decoded. After that if the instruction has an effective address in it the appropriate subroutines will decode that part of the instruction.

Each of the decoding subroutines uses the HEX_TO_CHAR subroutine to convert the HEX characters to ASCII and adds the resulting decoded parts of the instructions to the buffer which points to A1 address register. Then the PRINT subroutine is called that actually prints the buffer with the decoded instruction.

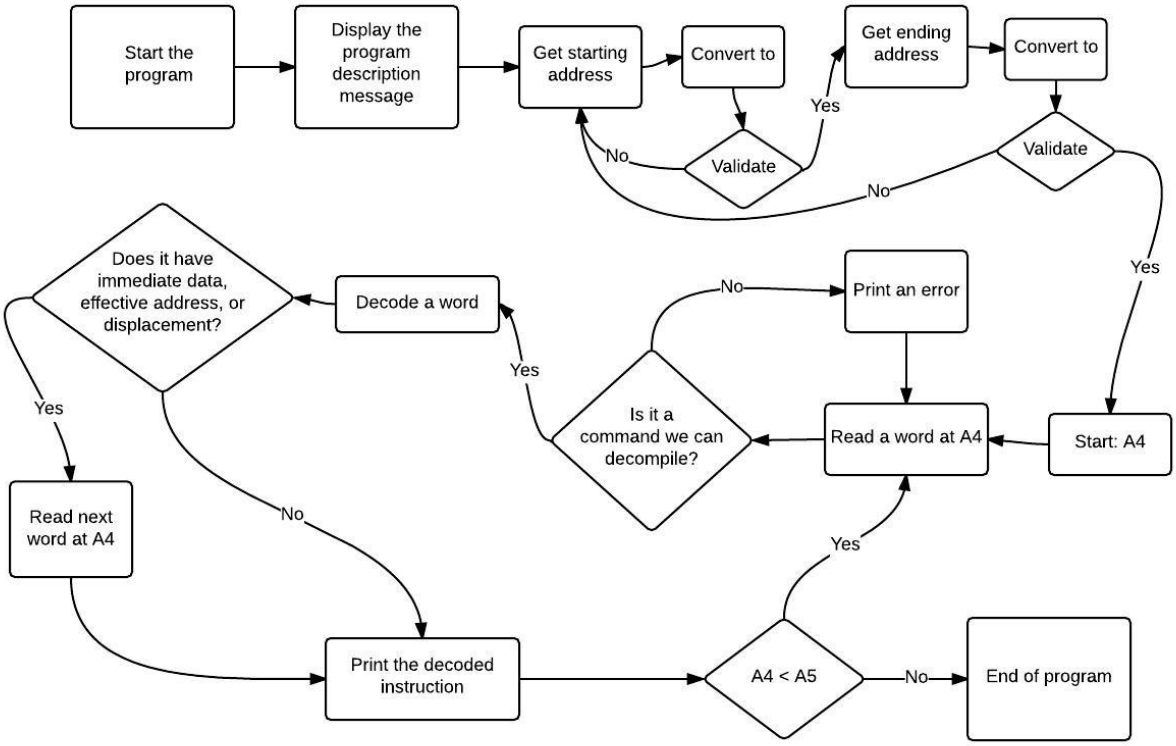

Figure 1 The flowchart of the program

Getting this project done especially OP codes on time was difficult but everything else seems pretty clever.