Contents

1 Introduction

2 Design

2.1 Part A

part a

2.2 Part B

partb

3 Testing

3.1 Part A

part a

3.2 Part B

partb

4 Questions

4.1 Question 1

"What happens when there is no exit code 0x0D provided in the initialization process? Would it cause a problem? Why or why not?" answer goes here

4.2 Question 2

"How can our code be modified to provide a variable address range? For example, what if I only wanted to convert the first 10 data entires? " answer goes here

5 Conclusion

conclusions

6 Appendix

6.1 Part A Assembler Code

```
/* DO NOT MODIFY THIS ------------------------------------------------------*/
.text

.global AssemblyProgram

AssemblyProgram:
lea       -40(%a7),%a7 /*Backing up data and address registers */
movem.l %d2-%d7/%a2-%a5,(%a7)
/*------------------------------------------------------------------------*/

/************************************************************/
/* General Information ****************************************/
/* File Name: Lab1a.s ****************************************/
/* Names of Students: Arun Woosaree and Navras Kamal
**/
/* Date: 1/29/2018
**/
/* General Description:
**/
/*
**/
/************************************************************/

/*Write your program here*************************************/

movea.l #0x2300000, %a1          /* save input address to a1*/
movea.l #0x2310000, %a2          /* save output address to a2*/

/* let a value in quotation marks be the ASCII value of the character enclosed

loop:                                        /* the looping function*/
 move.l (%a1), %d2                       /* move the value at address a1 to d2

 cmp.l #0x0D, %d2                        /* Check if the inval is the enter co
 beq end                                      /* if it is, go to the end of

 cmp.l #0x2F, %d2                        /* compare inval to the hex value of
 blt err                                      /* if inval is less than ASCI

 cmp.l #0x3A, %d2                        /* compare the inval to the hex value
 blt zeronine                            /* if it is less than the value of ":"
                                                  /*
thus go to the proper part of the code to handle this value*/

 cmp.l #0x41, %d2                        /* compare the inval to "A"*/
 blt err                                      /* if it is less than the "A"
```

```
 cmp.l #0x47, %d2                        /* compare the inval to "G"*/
 blt bigathruf                           /* if it is less than the value of "G
                                                                    /*
thus go to the part of the code to handle these values*/

 cmp.l #0x61, %d2                        /* compare the inval to "a"*/
 blt err                                         /* if it is in this range it

 cmp.l #0x67, %d2                        /* compare the inval to "g"*/
 blt littleathruf                        /* if it is less than "g" then it mus
                                                                    /*
thus go to the part of the code to handle these values*/

err:                                             /* if the inval is equal to o
 move.l #0xFFFFFFFF, (%a2)       /* throw the error code to the output address
 bra endloop                             /* go to the end of the loop before r

zeronine:                                        /* inval is between "0" and "
 sub.l #0x30, %d2                        /* subtract the hex value of "0" from
 move.l %d2, (%a2)                       /* move this calculted hex value to t
 bra endloop                             /* go to the end of the loop before r

bigathruf:                                       /* inval is between "A" and "
 sub.l #0x41, %d2                        /* subtracts the hex value of "A" d2.
 add.l #0xA, %d2                         /* adds the value of "A" to d2, which
 move.l %d2, (%a2)                       /* move this value to the output addr
 bra endloop                             /* go to the end of the loop before r

littleathruf:                            /* inval is between "a" and "f"*/
 sub.l #0x61, %d2                        /* subtracts the hex value of "a" d2.
 add.l #0xA, %d2                         /* adds the value of "a" to d2, which
 move.l %d2, (%a2)                       /* move this value to the output addr
 bra endloop                             /* go to the end of the loop before r

endloop:                                         /* handles code to be executed
 add.l #0x4, %a1                         /* increment the input address by 4*/
 add.l #0x4, %a2                         /* increment the output address by 4*
 bra loop                                        /* restart the loop*/

end:                                             /* end the custom part of the

/*End of program ************************************************/

/* DO NOT MODIFY THIS ------------------------------------------------*/
movem.l (%a7),%d2-%d7/%a2-%a5 /*Restore data and address registers */
```

```
lea        40(%a7),%a7
rts
/*-------------------------------------------------------------------------*/
```

6.2 Part B Assembler Code

```
/* DO NOT MODIFY THIS ------------------------------------------------------*/
.text

.global AssemblyProgram

AssemblyProgram:
lea        -40(%a7),%a7 /*Backing up data and address registers */
movem.l %d2-%d7/%a2-%a5,(%a7)
/*-------------------------------------------------------------------------*/

/*************************************************************/
/* General Information ***************************************/
/* File Name: Lab1a.s ****************************************/
/* Names of Students: Arun Woosaree and Navras Kamal
**/
/* Date: 1/29/2018
**/
/* General Description:
**/
/*
**/
/*************************************************************/

/*Write your program here*************************************/

movea.l #0x2300000, %a1          /* save input address to a1*/
movea.l #0x2320000, %a2          /* save output address to a2*/

/* let a value in quotation marks be the ASCII value of the character enclosed

loop:                                        /* the looping function*/
 move.l (%a1), %d2                        /* move the value at address a1 to d2

 cmp.l #0x0D, %d2                         /* Check if the inval is the enter co
 beq end                                     /* if it is, go to the end of

 cmp.l #0x41, %d2                         /* compare the inval to "A"*/
 blt err                                     /* if it is less than the "A"

 cmp.l #0x5B, %d2                         /* compare the inval to "["*/
```

```
 blt bigathruz                          /* if it is less than the value of "[
                                                  /*
thus go to the part of the code to handle these values*/

 cmp.l #0x61, %d2                       /* compare the inval to "a"*/
 blt err                                        /* if it is in this range it

 cmp.l #0x7B, %d2                       /* compare the inval to "{"*/
 blt littleathruz                       /* if it is less than "{" then it mus
                                                  /*
thus go to the part of the code to handle these values*/

 bigathruz:                                     /* inval is between "A" and "
 add.l #0x20, %d2                       /* adds the hex difference between "A
 move.l %d2, (%a2)                      /* move this value to the output addr
 bra endloop                            /* go to the end of the loop before r
 /*TODO*/
littleathruz:                           /* inval is between "a" and "z"*/
 sub.l #0x20, %d2                       /* subtracts the hex difference betwe
 move.l %d2, (%a2)                      /* move this value to the output addr
 bra endloop                            /* go to the end of the loop before r
 /*TODO*/
 err:                                           /* if the inval is not a valic
 move.l #0xFFFFFFFF, (%a2)      /* throw the error code to the output address
 bra endloop                            /* go to the end of the loop before r

 endloop:                                       /* handles code to be executed
 add.l #0x4, %a1                        /* increment the input address by 4*/
 add.l #0x4, %a2                        /* increment the output address by 4*
 bra loop                                       /* restart the loop*/

 end:


/*End of program ************************************************/

/* DO NOT MODIFY THIS ------------------------------------------------*/
movem.l (%a7),%d2-%d7/%a2-%a5 /*Restore data and address registers */
lea     40(%a7),%a7
rts
/*-----------------------------------------------------------------------*/
```