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(movem.l /*————————————————————————-*/
    /*****************************************************************/
/* 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, movea.l 0x2310000,
    /* let a value in quotation marks be the ASCII value of the character enclosed
by the quotation marks*/
    loop: /* the looping function*/ move.l (
    cmp.l 0x0D, beq end /* if it is, go to the end of the program (breaking the
loop)*/
    cmp.l 0x2F, blt err /* if inval is less than ASCII zero it is not valid, throw
an error*/
    cmp.l 0x3A, blt zeronine /* if it is less than the value of ":" then it must
be a value between "0" and "9"*/ /* thus go to the proper part of the code to
handle this value*/
    cmp.l 0x41, blt err /* if it is less than the "A" than it is invalid, throw an
error*/
    cmp.l 0x47, blt bigathruf /* if it is less than the value of "G" then it must
be in the range "A" through "F"*/ /* thus go to the part of the code to handle
these values*/
    cmp.l 0x61, blt err /* if it is in this range it is invalid, thus throw an error*/
    cmp.l 0x67, blt littleathruf /* if it is less than "g" then it must be in the
range "a" through "F"*/ /* thus go to the part of the code to handle these
values*/
    err: /* if the inval is equal to or above "g" then the code will naturally
continue here*/ move.l 0xFFFFFFFF, (bra endloop /* go to the end of the
loop before restarting the loop*/
    zeronine: /* inval is between "0" and "9"*/ sub.l 0x30, move.l bra endloop
/* go to the end of the loop before restarting the loop*/
    bigathruf: /* inval is between "A" and "F"*/ sub.l 0x41, add.l 0xA, move.l
bra endloop /* go to the end of the loop before restarting the loop*/
    littleathruf: /* inval is between "a" and "f"*/ sub.l 0x61, add.l 0xA, move.l
bra endloop /* go to the end of the loop before restarting the loop*/
    endloop: /* handles code to be executed before the start of a new loop*/
add.l 0x4, add.l 0x4, bra loop /* restart the loop*/
    end: /* end the custom part of the program*/
    /*End of program ****************************************************/
    /* DO NOT MODIFY THIS ————————————————————*/ movem.l
(lea 40(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 enc

    loop:                                                  /* the looping functio
     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 valu
                 /*     thus go to the part of the code to handle these valu

     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 valu

 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
 /*TODO*/
littleathruz:                                /* inval is between "a" and "
 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
 /*TODO*/
 err:                                                  /* if the inval is no
 move.l #0xFFFFFFFF, (%a2)  /* throw the error code to the output address
 bra endloop                                 /* go to the end of the loop

 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
/*-----------------------------------------------------------------------*/
```