Meera Krishnamoorthy

Wednesday 5PM Section

TA: Fabio Arai

Homework 2 Outline

Dec2String

Description: This function converts a 16-bit signed decimal value passed to it to a decimal representation of that value as an ASCII string. This is done by adding ‘0’ to each digit in the decimal value, and ‘null’ to the string after all the digits have been added to the string.The signed decimal value is the number ‘n’ passed to the function, and the hexadecimal representation of the decimal in ASCII should be stored in ‘a’, the second number passed to the function.

Operation: The function starts by checking if the number is negative by checking its first bit. If the first bit is set, the number is signed, and we negate that number and translate that result into an ASCII string. We also add a ‘-‘ to the string to represent that the number is negative. Then, we divide the number by the largest power of 10 possible and loops dividing the number by the power of 10. The quotient is a digit. ‘0’ is added to each digit to convert it to ASCII.

Arguments: AX – binary value to convert to decimal representation in ASCII, SI – address where the ASCII representation of 16 bit signed decimal value should be put

Return values: NONE

Local variables:

* arg: copy of passed value to convert (AX)
* pwr10: current power of 10 being computed (CX)
* digit: current digit necessary to convert to ascii(BX)

Shared variables: None

Global variables: None

Input: None

Output: None

Error Handling: None

Registers used: AX, SI, CX, BX

Algorithms: Repeatedly divide by powers of 10 to get each digit, add by 48 to convert to ascii value, and store in a.

Data Structures: None

Known Bugs: None

Limitations: Can only convert 16 bit digits, assumes valid address

Pseudocode

|  |  |
| --- | --- |
| Arg = n  pwr10 = 10000  Compare digit to 0  If (sign flag is set)  STORE (‘-‘, a)  a++  ENDIF  WHILE (Pwr10 > 0)  digit = arg/pwr10    Store remainder of arg/pwr10  STORE(digit + ‘0’, a)  a++  arg = arg % pwr10 ; gets remainder of arg  division by power of 10  pwr10 = pwr10 / 10    ENDWHILE  Add null character to address a | Copy n into AX (digit) (MOV AX, n)  Copy a into SI (MOV SI, a)  Set BX (result) to 0 (MOV BX, 0)  Set CX (pwr10) to 10000 (MOV CX, 10000)  Compare AX to 0 (CMP AX, 0)  Jump to negation label if AX is less than 0 (JCNegationLabel)  Jump to CalculateASCIIif AX is greater than 0 (JCXZ CalculateASCII)  Negation label:  Negate AX (NEG AX)  Move a dashed line into SI (MOV ‘-‘, [SI])  increment SI (INC SI)  CalculateASCII:  compare CX and 0 (CMP CX, 0)  Jump if less than or equal to to END (JLE END)  ;Jump if greater than to continue (;JG CalculateASCIIBody)  CalculateASCIIBody:  Make DX 0 (MOV DX, 0)  Divide AX by CX (DIV CX)  CMP AX, 0  JL ReducePower  Copy remainder of AX/CX, which is DX in BX (MOV BX, DX)  Move AX to SI (MOV word ptrAX, [SI])  ADD ‘0’ to SI (ADD [SI], ‘0’)  Increment SI (INC SI)  Copy CX into AX (MOV AX, CX)  Divide CX by 10 (DIV 10)  Move BX into AX (MOV AX, AX)  Compare CX to 1 (CMP CX, 1)  Jump to calculate ASCII if CX is greater than 1 (JGECalculateASCIIBody)  Jump to END if CX is not greater than 1(JL END)  ReducePower:  Move AX to BX (MOV BX, AX)  Move CX to AX (MOV AX, CX)  DIV 10  Move AX to CX (MOV CX, AX)  Move BX to AX (MOV AX, BX)  END:  ADD [SI], ‘null’ |

Hex2String

Description: This function converts a 16-bit signed decimal value passed to it to a hexadecimal representation of that value in ASCII. The signed decimal value is the number ‘n’ passed to the function, and the hexadecimal representation of the decimal in ASCII should be stored in ‘a’, the second number passed to the function. N is passed in AX by value, and a, the address, is passed in SI by value. If the first bit is set, the number is signed, and we will negate that number and translate that result into an ASCII string.

Operation: The function starts with the largest power of 16 (16^4) possible and loops dividing the number by the power of 16. The quotient is a hexadecimal digit. 48 is added to each digit to convert it to ASCII. If the hexadecimal digit is greater than or equal to 10, then 55 is added to it (65 is where A begins, but it is offset by 10, so 55 is added to digits greater than or equal to 10).

Arguments: AX – binary value to convert to hexadecimal representation in ASCII, SI – address where the ASCII representation of 16 bit signed decimal value should be put, CX – where pwr1

Return values: NONE

Local variables:

* arg: copy of passed binary value to convert
* pwr16: current power of 16 being computed
* digit: current digit necessary to convert to ascii

Shared variables: None

Global variables: None

Input: None

Output: None

Error Handling: None

Registers used: AX, SI

Algorithms: Repeatedly divide by powers of 16 to get each digit, add by 48 or 55to convert to ascii value, and store in a.

Data Structures: None

Known Bugs: None

Limitations: Can only convert 16 bit digits

Pseudocode

Arg = n

pwr16 = 65336

WHILE (Pwr16 > 0)

digit = int(arg/pwr16)

IF (digit < 10)

STORE(digit + ‘0’, a) ; converts digits to ASCII

ELSE

STORE(digit + ‘A’, a) ; converts letters to ASCII

ENDIF

a++

arg = arg % digit

pwr16 = pwr16 / 16

ENDWHILE

Add null character to address a

|  |  |
| --- | --- |
| Arg = n  pwr16 = 65536  WHILE (Pwr16 > 0)  digit = arg/pwr16    Store remainder of arg/pwr16  IF (digit < 10)  STORE(digit + ‘0’, a)  ELSE  STORE(digit + ‘A’, a)  ENDIF  arg = arg % pwr16 ; gets remainder of arg  division by power of 16  pwr16 = pwr16 / 16    ENDWHILE  Add null character to address a | Copy n into AX (digit) (MOV AX, n)  Copy a into SI (MOV SI, a)  Set BX (result) to 0 (MOV BX, 0)  Set CX (pwr10) to 10000 (MOV CX, 65536)  CalculateASCII:  Copy AX into BX (MOV BX, AX)  compare CX and 0 (CMP CX, 0)  Jump if less than or equal to to END (JLE END)  ;Jump if greater than to continue (;JG CalculateASCIIBody)  CalculateASCIIBody:  Make DX 0 (MOV DX, 0)  Copy BX into AX (MOV AX, BX)  Divide AX by CX (DIV CX)  CMP AX, 0  JL ReducePower  Copy remainder of AX/CX, which is DX in BX (MOV BX, DX)  Move AX to SI (MOV word ptr AX, [SI])  CMP SI, 10  Jump if SI is less than 10 to add ‘0’, (JL AddZero)  Jump if SI is greater than or equal to 10 to add ‘A’ (JGE AddLetters)  AddZero  ADD ‘0’ to SI (ADD [SI], ‘0’)  Increment SI (INC SI)  JMP ContinueCalculateASCIIBody  AddLetters  ADD ‘A’ to SI (ADD [SI], ‘A’)  Increment SI (INC SI)  ContinueCalculateASCIIBody:  Copy CX into AX (MOV AX, CX)  Divide CX by 16 (DIV 16)  Compare CX to 0 (CMP CX, 0)  Jump to calculate ASCII if CX is greater than 0 (JG CalculateASCIIBody)  Jump to END if CX is not greater than 0 (JLE END)  ReducePower:  Move AX to BX (MOV BX, AX)  Move CX to AX (MOV AX, CX)  Divide this by 16 (DIV 16)  Move AX to CX (MOV CX, AX)  Move BX to AX (MOV AX, BX)  END:  ADD [SI], ‘null’ |