Experiments No:6

Title: HEX to BCD and BCD to HEX

Problem Statement: Write X86/64 ALP to convert 4-digit Hex number into its equivalent BCD number and 5-digit BCD number into its equivalent HEX number. Make your program user friendly to accept the choice from user for: (a) HEX to BCD b) BCD to HEX (c) EXIT.Display proper strings to prompt the user while accepting the input and displaying the result. (Wherever necessary, use 64-bit registers).

Objective: To learn the implementation stack for its conversion of number

Outcomes: On completion of this practical ,students will be able to

C218.2: Apply logic to implement code conversion

Hardware Requirement: NA

Software Requirement: Ubuntu ,NASM etc.

Theory Contents :

1. Memory: model directives

These directives instruct the assembler as to how large the various segment (code, data, stack, etc) can be and what sort of segmentation register will be required.

.MODEL

<model >

Where <model>is one of the following options:

Tiny- code and data fit into single 64k and accessed via near pointers.

Small- code and segment both less than 64k and accessed via near pointers. Compact – Code segment is <64K (near ptr) and data segment is <1MB (far ptr) Medium- Code segment is <1Mb (far ptr) and data segment is <64K (near ptr) Large- code and data segment both less than 1MB and accessed via far pointers. Huge- Like “large” model, but also permits arrays larger then 64K

Segment directives

These directives indicate to assembler the order in which to load segment. When it encounter one of these directives, it interprets all subsequent instructor as belonging to the indicated segment (until the ne xt directives is encounter).

. data

.stack <size>; specified .code

Data type declaration

As has been previously discu ssed, data can be of several different lengths and assembler must be able to decide what length a specific constant (or variable) is. This can be down using data type declaration in conjunction with a constant declaration or variable assignment .this is akin to strong typing of variable in high level language .the data types are:

Byte (8 bit quantity)—synonyms are byte and db

Word (16) -- synonyms are word dw

Dword (32bit) -- synonyms are dword and dd

Qword (64bit) -- synonyms with dq

Tword (128bit) -- synonyms with dt An

example of their use is:

MOV AX, word VAR; moves a 16-bit Variable VAR into AX

2. Instruction Used:

1. PUSH:-Push word onto stack.
2. POP:-Pop word off stack.
3. DIV:-Divide byte/word
4. XOR: - Exclusive or byte/word
5. JA/JNBE:-Jump if above/not below or equal
6. JB/JNAE:-Jump if below /not above or equal
7. JG/JNLE:-Jump if /not less nor equal
8. JL/JNGE:-Jump if less /not greater nor equal
9. INT:-It allows ISR to be activated by programmer & external H\W device

3.New interrupt used:

1 INT 21h, function 0AH:- Read from keyboard and place into a memory buffer a row ofcharacter, until<CR>is pressed.

4..New directives used:

1. .MODEL
2. .STACK
3. .DATA
4. .CODE
5. .OFFSET: - It informs the assembler to determine the offset/displacement of a named data item.
6. .PTR: - assign a specific type to variable/label

5.Algorithm:

HEX to BCD

1. Define variable on data segment.
2. Display message on screen ENTER 4-DIGIT HEX NO.
3. Accept BCD NO from user.
4. Transfer 0AH as a divisor in one of the register.
5. Divide the no by 0AH
6. PUSH reminder in one of the register
7. Increment Count _1.
8. Repeat Till BCD NO is not zero go to step 5.
9. Pop the content of Reminder.
10. Display result by calling display procedure.
11. Decrement Count _1, till Count is not zero repeat step 9 else go to step 12.
12. Stop

BCD to HEX

1. Define variables in data segment
2. Display message on screen ENTER 5-DIGIT BCD NO.
3. Accept single digit from user
4. Transfer 10000 to multiplier
5. Multiply accepted BCD digit by multiplier & add it to RESULT variable.
6. Accept single digit from user
7. Transfer 1000 to multiplier
8. Multiply accepted BCD digit by Multiplier & Add it to RESULT variable.

9. Accept single digit from user
10. Transfer 100 to multiplier
11. Multiply accepted BCD digit by Multiplier & Add it to RESULT variable.
12. Accept single digit from user
13. Transfer 10 to multiplier
14. Multiply accepted BCD digit by multiplier & add it to RESULT variable.
15. Accept single digit from user
16. Transfer 1 to multiplier
17. Multiply accepted BCD digit by multiplier & add it to RESULT variable.
18. Display result by calling display procedure
19. Stop.

Procedure for accept numbers: (ASCII to HEX)

1. Read a single character/digit from keyboard using function 0AH of INT 21H
2. Convert ASCII to HEX as per following:

a. Compare its ASCII with 30H if No is less than 0 (i.e case of -ve no given) then go to step f else go to step c.

b. Compare its ASCII with 39H if No is greater than 9 (i.e case of character A – F given) then go to step f else go to step c .

c. Store the resultant bit in NUM Variable.

d. Check whether four digits (16-bit number) or two digits (8-bit number) are read; if yes then go to display procedure else go to step 1 for next bit

e. Till counter is zero go to accept procedure.

f. Display massage “I/P is invalid BCD number” & go to step 17.

3. End of accept procedure.

Procedure for display Result: (HEX to ASCII)

1. Compare 4 bits (one digit) of number with 9
2. If it is <= 9 then go to step 4 else go to step 3
3. Add 07 to that number
4. Add 30 to it
5. Display character on screen using function 02 of INT 21H
6. Return to main routine
7. End of display procedure.

HEX to BCD

Divide FFFF by 10 --- note this FFFF is as decimal 65535 so

Division

65535 / 10 Quotient = 6553 Reminder = 5

6553 / 10 Quotient = 655 Reminder = 3

655 / 10 Quotient = 65 Reminder = 5

65 / 10 Quotient = 6 Reminder = 5

6 / 10 Quotient = 0 Reminder = 6

and we are pushing Reminder on stack and then printing it in reverse order

BCD to HEX

1 LOOP : DL = 06 ; RAX = RAX * RBX = 0 ; RAX = RAX + RDX = 06

2 LOOP : DL = 05 ; 60 = 06 * 10 ; 65 = 60 + 5

3 LOOP : DL = 05 ; 650 = 60 * 10 ; 655 = 650 + 5

4 LOOP : DL = 03 ; 6550 = 655 * 10 ; 6553 = 6550 + 3

5 LOOP : DL = 06 ; 65530 = 6553 * 10 ; 65535 = 65530 + 5

Hence final result is in EAX = 65535 which is 1111 1111 1111 1111 and when we print this it is represented as FFFF

Assignment Questions:

1. Explain various addressing modes of 8086 microprocessor used in this program.
2. Explain Different assembler directives used in this program.
3. Explain various Number Systems used in Digital Electronics.
4. What is HEX Number? Explain the s teps to convert HEX No to BCD No
5. What is BCD Number? Explain the steps to convert BCD No to HEX No.

Conclusion: In this way we studied about hex to bcd and hex to BCD number conversion.

MPL Practical Oral Question Bank

Sr No	B L	Questions	Oral 1	Oral 2 (improvement)	Remark
1	2	Explain the use of SI register in the program.			
2	2	Explain the use of DI register in the program.			
3	1	Is there any difference between Rotate Bit & Shift Bit Instruction?			
4	2	Explain the difference 32 bit instruction and 64 bit instruction?			
5	1	What is Push,Pop and Div instruction?			

Sign of Student