Brandon london 2700 homework 5

Brandon London 2700 Homework 5

12.1)

a) $23_{dec} = (00100011)_{bin}$

Convert $(00100011)_{bin}$ into hexadecimal

$(00100011)_{bin} = 23$ hex

Packed decimal format of 23 in hex notation is $\boxed{23}$

b) ASCII characters 23 in hex notation are $\boxed{32\ 33}$

ASCII 2 $48+2=50$

ASCII 3 $48+3=51$

→ ASCII 23 in decimal is 50 51

16 | 50

3 - 2

$50_{dec} = 32_{hex}$

16 | 51

3 - 3

$51_{dec} = 33$ hex

12.2)

Used chart for Answers

a) 0111 0011 0000 1001

Packed decimal number is $\boxed{7309}$

| A | B | C | D | output |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 2 |
| 0 | 0 | 1 | 1 | 3 |
| 0 | 1 | 0 | 0 | 4 |
| 0 | 1 | 0 | 1 | 5 |
| 0 | 1 | 1 | 0 | 6 |
| 0 | 1 | 1 | 1 | 7 |
| 1 | 0 | 0 | 0 | 8 |
| 1 | 0 | 0 | 1 | 9 |

b) 0101 1000 0010 $\boxed{582}$

c) 0100 1010 0110

4 ? 6

1010 is not a valid packed decimal numer and resuts in error

13.2) a)

$\boxed{x3 = x2}$

b) $\boxed{x_3 = (x2)}$

c) Effective address = PC + offset + 1
$x_3 = x_1 + x_2 + 1$

d) $x_3 = x_2 + x_4$

Relationship between various addressing modes is shown above.

13.3) a) The address of the operand remains in the instruction itself as immediate value. Thus, in immediate addressing mode, effective address of operand is $\boxed{14}$

b) Address of the operand with direct addressing mode is $\boxed{\text{memory location of 14}}$

c) in indirect addressing, address of the data is held in intermediate location.
- first checks for the address and then used to locate and identify the data.

EA=(A)

* effective address of the operand is memory location whos address is in memory location 14 that is stored in the address X

d) address of the operand using register addressing mode is $\boxed{\text{Register } 14}$

e) memory location whose address is in register 14

One Address Instruction:

Program to compute x = (A+3×C)/(D-E×F) ~~f~~ using on address:
Load E; Load value of E mul F; multiply value of E with F store R; Store
result in R load D; load value D sub R; subtract result R with D store R; Store
result of (D-E×F) D-E × F in R load B; load value of B mul C; multiply B with
C ADD A; Add result with A Div R; Divide R with A store X; store result in X.

- One address used registers for data manipulation purpose.
- Addition and subtraction instruction uses registers.
- Division and multiplication doesn't need a second register. However its assumed Accumulator contains result of the operation

Two Address Instruction: Program to compute X = (A+3×C)/(D-E×F) using Two address

move R1, E; move E to register R1
mul R1, F; ~~the~~ contents of R1 (E) is multiplied with F and store result in R1
move R2; D move D to register R2
sub R2, R1; subtract content R1 with R2 (D) and store result in R2
move R1, B; move B to register R1
mul R1, C; multiple register R1 (B) with C and store result in R1
Add R1, A; Add R1 (C)
Div R1, R2; Divide R1 with R2 and store the results in R1.
move X, R1; finally store content of R1 to X

- The most common instruction in the assembly language used is two address ~~[illegible]~~ Instructions. each address feild in the instruction specifies
  * register * memory word
- Move instruction in two address moves the data to and from memory locations to registers

Three Address Next page

12.4)

1698 - 0001 0110 1001 1000
1786 - 0001 0111 1000 0110

| | | | |
|---|---|---|---|
| 0010 | 1101 c=1 | 0001 | 1110 |
| d1 | $d_2$ | $d_3$ | $d_4$ |
| | | | 0110 |

$c_1$=1 0100

10010
0110

$c_2$=1 1000
~~0110~~

0001 0110 1001 1000
0001 0111 1000 0110
0010 1101 10001 1110
$c_4$=1 0110 $c_1$=1 0110

$\boxed{0011}$ $c_3$=1 0011 c1=1 $\boxed{0100}$
$c_2$=1 10010
0110

$3484$ c4=1 $\boxed{0100}$ $c_2$=1 $\boxed{1000}$

final result = $\boxed{0011\ 0100\ 1000\ 0100}$

12.6) Zero addeness instruction program to compute $x = (A+B\times C)/(D-E\times F)$

Using zero address
push A; Push A to stack
Push B; Push B to stack
push C; PushC to stack
mul; multiply B with C
Add; Add result with A
PushD; Push d to stack
Push E; Push E to stack
pushF; PushF to stack
mul; multiply E with F
sub; subtract result with D
Div; divide resut of add with subtract result
Popx; Pop value from the stack

- Zero Address instructions are the instructions organized to stack in which it not used addresses fields for the above said instructions (Add, mul, Div, SUB, Push Pop.
- Operand sources and destination are both implicit in zero address instruction.
- Address of opperand will be in special register which is automatically incremented or decremented.

for expression $(A+B\times C)$
first we are moving A to top of the stack. Then B is push to top replacing A then C is move to top of stack replacing B. now multiply B with C and store result.

for expression $(D-E\times F)$
D is first moved "D" to top of the stack, then E to top of the stack and F to top of stack. Now the content of top of the stack is F. multiply top of stack F and E and store result. Subtract d and store after deletion.

12.6) Three address instruction: Program to compute $X = (A+B\times C)/(D-E\times F)$ using three address:

MUL R1, E, F; multiply E and F and store in R1 ~~& multiply E and F and~~

SUB R1, D, R1; Subtract R1 ($E\times F$) with D and store in R1

MUL R2, B, C; multiply B and C and store result in R2

ADD R2, A, R2; Add R2 ($B\times C$) with A and store result in R2

DIV, R1, R2; Divide R1 by $R_2$ and store result in X

• Every instruction in three address instruction used processor register or memory operands

* Three Address instruction takes only 5 instructions to evaluate the given expression where as two address takes 9 instructions, one takes 11 instructions and zero address takes 12 instructions.

13.1) a) 13.1 - 13.3

instruction itself contains immediate value 20

operand = 20

Thus, load immediate 30 results with data $\boxed{20}$

b) load direct 20 results with fetching data $\boxed{40}$

c) in indirect addressing, address of the data is held in intermediate location

word 20 contains 40

Thus, load indirect 20 results with fetching data $\boxed{60}$

d) operand = 30

Load immediate 30 results with data $\boxed{30}$

e) Load direct 30 results with fetching data $\boxed{50}$

F. load indirect 30 results in fetching data $\boxed{70}$

Brandon London 2700 Homework 5

12.1)

a) $23_{dec} = (00100011)_{bin}$

Convert $(00100011)_{bin}$ into hexadecimal

$(00100011)_{bin} = 23$ hex

packed decimal format of 23 in hexa notation is $\boxed{23}$

B) ASCII characters 23 in hexa notation are $\boxed{32\ 33}$

ASCII 2 48+2=50

ASCII 3 48+3=51 → ASCII 23 in decimal is 50 51

16|50 16|51

3-2 3-3 $51_{dec}$ = 33 hex

$50_{dec} = 32_{hex}$

12.2)

Used chart for Answers

a) 0111 0011 0000 1001

packed decimal number is $\boxed{7309}$

| A | B | C | D | output |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 2 |
| 0 | 0 | 1 | 1 | 3 |
| 0 | 1 | 0 | 0 | 4 |
| 0 | 1 | 0 | 1 | 5 |
| 0 | 1 | 1 | 0 | 6 |
| 0 | 1 | 1 | 1 | 7 |
| 1 | 0 | 0 | 0 | 8 |
| 1 | 0 | 0 | 1 | 9 |

B) 0101 1000 0010 $\boxed{582}$

C) 0100 1010 0110

4 ? 6

1010 is not a valid packed decimal numbr and results in error

12.3) a)

Smallest integer for unsigned is $\boxed{0}$ and largest is $\boxed{255}$

$2^8 = 256$ 0-255

b) Smallest integer for sign-magnitude is $\boxed{\text{"-127"}}$ and largest for sign mag is $\boxed{127}$

| | | | |
|---|---|---|---|
| 0 | Positive | 0000000 (0) | 1111111 (127) |
| 1 | Negitive | 0000000 (-0) | 1111111 (-127) |

c) Smallest integer for ones compliment is $\boxed{-127}$ and largest integer is ~~128~~ $\boxed{127}$

complement range $= -(2^{n-1} - 1)$ to $(2^{n-1} - 1)$

$-(2^7 - 1)$ to $(2^7 - 1)$

$= -127$ to $127$

d) Smallest integer for two's compliment is $\boxed{-128}$ and largest is 127

$-(2^{n-1})$ to $(2^{n-1} - 9)$

$-(2^7)$ to $(2^7 - 1)$

$\boxed{-128 \text{ to } 127}$

e) Smallest unsigned Packed decimal integer is $\boxed{0}$ and largest unsigned packed decimal is $\boxed{99}$

| | Nibble 2 | Decimal | Nibble 1 | decimal |
|---|---|---|---|---|
| min | 0000 | 0 | 0000 | 0 |
| max | 1001 | 9 | 1001 | 9 |