第２回　数の表現（２）　予習資料

第2回授業の前にあらかじめ目を通し、問題を解いておくこと。

［１］　（２進⇒１０進）変換の復習　　（アーキテクチャの教科書にはこの内容の記述は無い）

a.　1101_2 ＝（　　）$_{10}$　　100110_2 ＝（　　）$_{10}$

　　1101101_2 ＝（　　）$_{10}$　　11110000_2 ＝（　　）$_{10}$

b.　2^X を覚えてしまおう

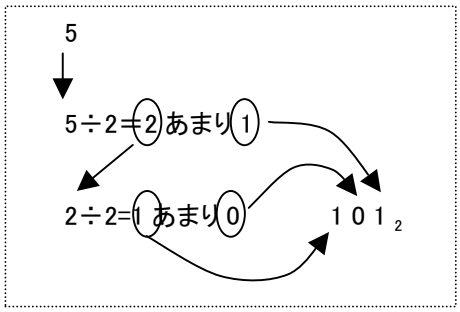
① 2^2 =（　）　② 2^3 =（　）　③ 2^4 =（　）　④ 2^5 =（　）　⑤ 2^6 =（　）　⑥ 2^7 =（　）

⑦ 2^8 =（　）　⑧ 2^9 =（　）　⑨ 2^{10} =（　）　ここまでは順番に唱える

⑩ 2^{12} =（　）　⑪ 2^{15} =（　）　⑫ 2^{16} =（　）　この3つはよく出てくるので覚える

［２］　（１０進⇒２進）変換　　（アーキテクチャの教科書にはこの内容の記述は無い）

10 進数の 5_{10} を、2 進数に変換するのに、1つの方法は、右図のように繰返して 2 で割って余りを下から並べていく方法がある。 5 を 2 で割って 5÷2＝2 あまり 1、商の 2 を 2 で割って 2÷2=1 あまり 0。 これらを並べて、101 とする。



a.　次の１０進数を２進数に変換せよ

① 3_{10} ⇒（　　）$_2$　　② 6_{10} ⇒（　　）$_2$

この、次々と繰返して2で割る方法は、桁数が大きくなると結構大変である。

b.　次の１０進数を２進数に変換せよ

① 7_{10} ⇒（　　）$_2$　　② 15_{10} ⇒（　　）$_2$

1つのトリックとして、2^N を覚えておいて、その周辺の数は 2^N から調整して作る方法がある。たとえば、5_{10} は 4_{10} の「次」であるから、$4_{10}=2^2=100_2$ を覚えていれば、100_2 の次（100＋1）の数は 101_2 であることは、簡単に求まる。 同じように、3_{10} は 4_{10} ＝100_2 の「手前」であるから、100_2 の手前（100_2－1）の数は 11_2 である（11_2 の次が 100_2）。

c.　次の１０進数を２進数に変換せよ （慣れるために、2進数は 16 桁の数として書け。上位数字は0でよい）

① 7_{10} ⇒（　　）$_2$　　② 15_{10} ⇒（　　）$_2$

③ 30_{10} ⇒（　　）$_2$　32 の2つ手前　　④ 254_{10} ⇒（　　）$_2$　256 の2つ手前

⑤ 1020_{10} ⇒（　　）$_2$　1024 の4つ手前

⑥ 645_{10} ⇒（　　）$_2$　これはどうしようもない。645＝512＋133＝512＋128＋5

［３］　１６進数の活用　（アーキテクチャの教科書にはこの内容の記述は無い）

2進数は、桁数が多くなり表記も面倒であるし、2を掛けたり、2で割ったりする回数がとても多くなる。これの手を抜く方法として、16進数を使うことがあるので、紹介する。 まず16進数の基本から始めよう。

a.　16進数の扱いは、N進数でN＝16とするだけのことである。 但し、16進数の1桁を表すのに、0～9では足りない（文字の種類が 10 種類しかないが、16 進表記だと 16 種類必要）になる。 そこで、 0～9の後ろに、 A(=10)、 B(=11)、 C(=12)、 D(=13)、 E(=14)、 F(=15) の6文字を追加する。

つまり、16進数のそれぞれの桁は、0、1、2、…、9、（　　）のいずれかである。

16進数を10進数に変換するには、たとえば、

ABC_{16} ＝ $10\times16^2+11\times16^1+12\times16^0$ ＝ $10\times256+11\times16+12$ ＝（　　）＋（　　）＋12 ＝ 2748 となる。

b. 次の16進数を10進数に変換せよ （符号無しとする。符号の話は後から出てくる）

① $000D_{16}$ ⇒ （　　）$_{10}$　② $005A_{16}$ ⇒ （　　）$_{10}$　③ 3421_{16} ⇒ （　　）$_{10}$

④ $00FF_{16}$ ⇒ （　　）$_{10}$　⑤ $0FFF_{16}$ ⇒ （　　）$_{10}$　⑥ $FFFF_{16}$ ⇒ （　　）$_{10}$

⑦ $FF00_{16}$ ⇒ （　　）$_{10}$　⑧ $ED68_{16}$ ⇒ （　　）$_{10}$

（注） 実は④～⑥は、前のページにある「1つ手前」を使って楽に計算できる。

④ $00FF_{16}$ は 0100_{16} の「1つ手前」である。（F の次が 10 だから。 イメージとしては 0099 の次が 0100）
そこで代わりに、0100_{16} を考え、0100_{16} ＝（　　）$_{10}$ その1つ手前だから、$00FF_{16}$ ⇒ （　　）$_{10}$

⑤ これも同様に、$0FFF_{16}$ は 1000_{16} の「1つ手前」である。 1000_{16} ＝（　　）$_{10}$ なので、
その1つ手前だから、 $0FFF_{16}$ ⇒ （　　）$_{10}$

⑥ これも同様に、$FFFF_{16}$ は 10000_{16} の「1つ手前」である。 10000_{16} ＝（　　）$_{10}$ なので、
その1つ手前だから、 $FFFF_{16}$ ⇒ （　　）$_{10}$

16進数と2進数との間で、簡単に変換できる性質がある。 たとえば、

$1001\ \ 0110\ \ 1110\ \ 0011_2 = 96E3_{16}$

のように、2進数を4桁ずつ区切って、それぞれの4桁を16進数で読む、という方法で読み換えることができる。（なぜか？）

c. 次の2進数を16進数に変換せよ （符号無しとする）

① $0000\ \ 1000\ \ 0000\ \ 1011_2$ ⇒（　　）$_{16}$　② $0000\ \ 1111\ \ 1111\ \ 1111_2$ ⇒（　　）$_{16}$

③ $1010\ \ 1100\ \ 1110\ \ 0010_2$ ⇒（　　）$_{16}$　④ $1111\ \ 1111\ \ 1111\ \ 1111_2$ ⇒（　　）$_{16}$

逆向きの変換も同じようにできる。

d. 次の16進数を2進数に変換せよ （符号無しとする）

① $ACBD_{16}$ ⇒ （　　）$_2$　② $1F5B_{16}$ ⇒ （　　）$_2$

③ $ED68_{16}$ ⇒ （　　）$_2$　④ $C31F_{16}$ ⇒ （　　）$_2$

$1010_2=10_{10}=A_{16}$、 $1011_2=11_{10}=B_{16}$、 $1100_2=12_{10}=C_{16}$、 $1101_2=13_{10}=D_{16}$、 $1110_2=14_{10}=E_{16}$、 $1111_2=15_{10}=F_{16}$は覚えよ

e. なぜ、16進⇔2進の変換が2進を4桁ずつ切るだけでいいのか？ 下記の式をヒントにして説明せよ。

$abcd\ efgh_2 = a\times2^7+b\times2^6+c\times2^5+d\times2^4 + e\times2^3+f\times2^2+g\times2^1+h\times2^0$

$= (a\times2^3+b\times2^2+c\times2^1+d\times2^0)\times2^4 + (e\times2^3+f\times2^2+g\times2^1+h\times2^0)$

2進数で計算するよりは、16進数の方が桁の数が少ないので、大分に楽だろう。 たとえば、

$0110\ \ 1001\ \ 1110\ \ 1101_2 = 69ED_{16} = (6x4096 + 9x256 + 14x16 + 13)_{10} = 149997_{10}$となる。

f. では、もう一度2進⇔16進の変換を復習せよ

① $ADCB_{16}$ ⇒ （　　）$_2$　② $6B2F_{16}$ ⇒ （　　）$_2$

③ 1234_{16} ⇒ （　　）$_2$　④ $BC1A_{16}$ ⇒ （　　）$_2$

⑤ $0011\ 1100\ 0000\ 1011_2$ ⇒（　　）$_{16}$　⑥ $1111\ 0011\ 1100\ 0000_2$ ⇒（　　）$_{16}$

⑦ $0000\ 0000\ 0000\ 1010_2$ ⇒（　　）$_{16}$　⑧ $1010\ 1011\ 1100\ 1101_2$ ⇒（　　）$_{16}$

16進数の知識を使って、2進⇒10進変換を工夫してみよ。2進を16進に変換し、次に16進⇒10進の変換をする。

g. 次の2進数を10進数に変換せよ （符号無しとする）

① $0000\ 1000\ 0000\ 1011_2$ ⇒（　　）$_{10}$　② $0000\ 1111\ 1111\ 1111_2$ ⇒（　　）$_{10}$

③ $1010\ 1100\ 1110\ 0010_2$ ⇒（　　）$_{10}$　④ $1111\ 1111\ 1111\ 1111_2$ =（　　）$_{10}$

16進数の知識を使って、10進⇒2進変換を工夫してみよ。 10進⇒16進変換をし（16で繰り返し割る）、その結果を2進にしてみよ。

h. 次の10進数を2進数に変換せよ （符号無し、全て正の整数、符号については後から出てくる）

① 7_{10} = ( )$_2$ ② 10_{10} = ( )$_2$ ③ 13_{10} = ( )$_2$

④ 15_{10} = ( )$_2$ ⑤ 16_{10} = ( )$_2$ ⑥ 26_{10} = ( )$_2$

⑦ 80_{10} = ( )$_2$ ⑧ 79_{10} = ( )$_2$ ⑨ 128_{10} = ( )$_2$

⑩ 127_{10} = ( )$_2$ ⑪ 256_{10} = ( )$_2$ ⑫ 255_{10} = ( )$_2$

⑬ 1023_{10} = ( )$_2$ ⑭ 4095_{10} = ( )$_2$

⑮ 65535_{10} = ( )$_2$ ⑯ 65536_{10} = ( )$_2$

［４］ ２進数上での足し算・１６進数上での足し算 （アーキテクチャの教科書にはこの内容の記述は無い）

a. ついでに、2進数での足し算（筆算）ができるようになっておくとよい。次の2進数の足し算を筆算でせよ。（符号無しとする）

① $0000\ 0101_2 + 1000\ 0001_2$ = ( )$_2$ ② $0000\ 1101_2 + 0011\ 0110_2$ = ( )$_2$

③ $0011\ 1101_2 + 1010\ 1010_2$ = ( )$_2$ ④ $0111\ 1111_2 + 1111\ 1110_2$ = ( )$_2$

⑤ $0110\ 1001\ 0111\ 1111_2 + 0000\ 0110\ 1001\ 0001_2$ = ( )$_2$

⑥ $0111\ 0000\ 0011\ 1010_2 + 1111\ 1011\ 1111\ 1110_2$ = ( )$_2$

（理論の復習）

この足し算は、2進数のまま計算できるので、慣れておくこと。 計算の仕方は10進数のときと同じで、下の桁から、桁ごとに足してゆき、繰上りが出れば上の桁に加える。

まずは、慣れている 10進 で、どうやって計算しているか（手順を）考えてみよう。 36 + 27 を計算するのに、1桁目の 6 と 7 を足して 13、つまり1桁目の結果は 3 であり、繰上りとして1がある。次に2桁目の 3 と 2 を足すのだが、それに加えて1桁目からの繰上り 1 を足して、3+2+1 で 6 になる。だから結果は、63。

これと同じことを、2進数で行う。但し、1桁の足し算は、0+0=0, 0+1=1, 1+0=1, 1+1=10 （1桁目の結果は 0 で、繰上りが 1）、である。

たとえば、$11_2 + 10_2$ を計算するとき、最も下の桁（第1桁目）は 1+0 で 1 になる。その上の桁（第2桁目）は 1+1 で 10 になる。だから全体では、$11_2 + 10_2 = 101_2$ となる。 これは10進数に変換すれば 3+2=5 である。

同様に、$111_2 + 001_2$を計算すると、下の桁から、1桁目が 1+1=10、2桁目が 1+0 に繰上がりの 1 を足すから 1+0+1=10、3桁目も2桁目と同じで 1+0+繰上り 1=10。 全体を並べると $111_2 + 001_2 = 1000_2$となる。 この繰り返しで、16桁や32桁の足し算ができる。

b. 同様に、16進数での足し算（筆算）もできるようになっておくとよい。（符号無しとする、符号については後から出てくる）特に2進数で桁数が多い場合（16桁とか32桁とか）には、16進数で読み替えて足し算をしその結果を2進数に読み替える方が、2進数で何回も繰上りを考えるのに比べて、間違いが少ないだろうと思う。

① $3867_{16} + 5899_{16}$ = ( )$_{16}$ ② $325B_{16} + 22E9_{16}$ = ( )$_{16}$

③ $762F_{16} + 58D3_{16}$ = ( )$_{16}$ ④ $D6FA_{16} + 6221_{16}$ = ( )$_{16}$

⑤ 0022_{16} +$ACFB_{16}$ = ( )$_{16}$ ⑥ $2124_{16} + CF01_{16}$ = ( )$_{16}$

⑦ $3FFF_{16} + 0001_{16}$ = ( )$_{16}$ ⑧ $7425_{16} + 0FFF_{16}$ = ( )$_{16}$

⑨ $9898_{16} + FFFF_{16}$ = ( )$_{16}$

（理論の復習）

16進1桁の足し算のルールを理解する必要がある。

①入力が0～9までの場合は10進と同じ足し算をするが、結果が10を越えたところで16進表記を使わなければならない。たとえば6_{16}=6_{10}、7_{16}=7_{10}なので6_{16}＋7_{16}は13_{10}だが16進表記だとD_{16}となる。足した結果が16を超えると繰上りが発生

する。たとえば$9+9=18_{10}$だが繰上がるので12_{16}(2_{16}繰上り1_{16})である。
②入力がA_{16}～F_{16}の場合は足し算ルールを覚えてもよいし(たとえば$A_{16}+1_{16}=B_{16}$や、$A_{16}+6_{16}=10_{16}$など)、毎回10進で計算してもよい。16 進 1 桁であれば、それを 10 進に変換するのも和を 16 進に戻すのも、たいした手間ではない。

［５］　「２の補数」による負数の表現　（教科書 5.1.2）

教科書 5.1.2 を読んで理解する。

＜ことば＞

最上位ビット（ＭＳＢ）とは ……

「符号と絶対値表現」（通称「絶対値表現」）とは ……

……

１の補数とは ……

２の補数とは ……

2の補数は、負の整数を表す方法(の1つ)で、次のような作り方で符号を反転する。

1の補数の作り方: 絶対値「5」を2進で表現 ⇒ すべてのビットで 0/1 を反転

2の補数の作り方: まず始めに1の補数を作る(つまりすべてのビットで 0/1 反転) ⇒ その結果に「1」を加える

このとき、全体がNビットとすると、最上位(左端)のビットは「符号ビット」となり、正の数を表すのに使えるのは N-1 ビットになる。たとえば、全体を16ビットとすると、正の数を表すのに使えるのは15ビットである。

負数を作る例: $5_{10}=101_2$を符号反転して、-5_{10}を作りたいとする。 まず $5_{10}=101_2$を16ビットとすると、0000 0000 0000 0101 であるから、そのビットの 0/1 を反転すると、1111 1111 1111 1010 である。 次にこれに1を加えるので、1111 1111 1111 1011 となる。これが-5_{10}の「(全体を16ビットとしたときの)2の補数」表現である。

全体のビット数として違うものを用いると、たとえば全体が8ビットしかないとすると、$5_{10}=0000\ 0101_2$であるから、その 0/1 を反転すると 1111 1010 となり、それに1を加えると、1111 1011 となる。

また、全体が32ビットであれば、$5_{10}=0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0101_2$であるから、その 0/1 を反転して1を加えると、1111 1111 1111 1111 1111 1111 1111 1011 となる。

a. 下記の10進数を、2進の2の補数で表せ。但し、16ビットで表すとする。

① -7_{10} ⇒ (　　)$_2$　② -15_{10} ⇒ (　　)$_2$

③ -254_{10} ⇒ (　　)$_2$　④ -1024_{10} ⇒ (　　)$_2$

⑤ -1_{10} ⇒ (　　)$_2$　⑥ -2_{10} ⇒ (　　)$_2$

b. 下記の10進数を、2進の2の補数で表せ。但し、32ビットで表すとする。

① -7_{10} ⇒ (　　)$_2$

② -15_{10} ⇒ (　　)$_2$

③ -254_{10} ⇒ (　　)$_2$

④ -1024_{10} ⇒ (　　)$_2$

⑤ -1_{10} ⇒ (　　)$_2$

⑥ -2_{10} ⇒ (　　)$_2$

c. 10進数の(-5)を、①絶対値表現、②1の補数表現、③2の補数表現、で表現せよ。 但しいずれも、16 ビットで表すとする。

（注） 絶対値表現は、15ビットの絶対値 5 ＝ 000 0000 0000 0101 の左端(最上位)に負を表す符号ビット 1 を付け加えて作る。

d. 最上位のビット(一番左側のビット)を、「符号ビット」と呼ぶ事がある。(①絶対値表現、②1の補数表現、③2の補数表現)について、「-5」の最上位ビットがどうなっているか(1 か 0 か)。 ④また、正の数「5」(＝「+5」)の最上位ビットはどうか？

e. 下記の2進数を、16ビットの2の補数を表していると見て、10進数に変換せよ。(2進⇒10進変換)

① $0000\ 0000\ 0001\ 1010_2$ ⇒ (　　)$_{10}$　② $1111\ 1111\ 1111\ 1010_2$ ⇒ (　　)$_{10}$

③ $0000\ 0001\ 1010\ 0000_2$ ⇒ (　　)$_{10}$　④ $1111\ 1111\ 1010\ 1010_2$ ⇒ (　　)$_{10}$

(注)符号つきの2進数を10進に変換するには、(A)符号が正を表すならそのまま2進⇒10進変換し、(B)符号が負を表すなら、まず符号を反転し(つまりビット反転をしてから1を足す)、その結果を2進⇒10進変換し、得られた10進数の符号を反転する(マイナスをつける) 符号ビットは最上位(最左端)のビットである。

例: $0000\ 0001\ 0010\ 0011_2$ ⇒ 符号が正 ⇒ そのまま2進10進変換 ⇒ 256+32+3 =(291)$_{10}$

$1111\ 1111\ 1110\ 0111_2$ ⇒ 符号が負 ⇒ 補数による符号反転 ⇒ $0000\ 0000\ 0001\ 1001 = 25_{10}$ ⇒ —を付け -25_{10}

f. 下記の16進数を、16ビットの2の補数を表していると見て、10進数に変換せよ。

(注) 16進数でも、それを2進数に置きなおして、上と同じ手続でできる。

① $000D_{16}$ ⇒ (　　)$_{10}$　② $005A_{16}$ ⇒ (　　)$_{10}$

③ 3421_{16} ⇒ (　　)$_{10}$　④ $00FF_{16}$ ⇒ (　　)$_{10}$

⑤ $0FFF_{16}$ ⇒ (　　)$_{10}$　⑥ $FFFF_{16}$ ⇒ (　　)$_{10}$

⑦ $FF00_{16}$ ⇒ (　　)$_{10}$　⑧ $ED68_{16}$ ⇒ (　　)$_{10}$

例: $FEDC_{16}$ ⇒ 符号が負 ⇒ 2の補数として符号反転 ⇒ 0000 0001 0010 0100 ⇒ 2進10進変換 ⇒

256+32+4 ⇒ 292 ⇒ 再度符号反転して符号を元に戻す ⇒ (−292)$_{10}$

［6］　1の補数表現の問題点　　(アーキテクチャの教科書にはこの内容の記述は無い)

a. 値0を表現せよ(　①　)。 次に、16ビットの1の補数で符号を反転する(つまり-0)とどうなるか(　②　)。本来 −0 ＝ 0 のはずであるが、どうか？

b. 同じことを2の補数で考える。まず値 0 を表し(①)、次に16ビットの2の補数で符号を反転するとどうなるか(②)。 1の補数の場合と何が違うか述べよ。

［7］　足し算と２の補数の関係　　(アーキテクチャの教科書にはこの内容の記述は無い)

a. 8ビット(符号1ビット＋7ビット)の2の補数表現で、 $5_{10}+(-3_{10})$ を筆算形式に書いてみよ。

それを、それぞれを8ビットの符号無しの2進数(正の2進数)だと思って、2進数の筆算で足すとどうなるか。

b. 上記と同じやり方で、8ビットの2の補数表現で、次の計算をせよ。つまり引く数を2の補数として符号反転し、「引く」代りに「負数を足す」形で計算せよ (筆算で行え)

① 01111110_2—00001011_2

② 01001010_2—00000011_2

③ 01110111_2—00001111_2

④ 01111111_2—00000011_2

＜おまけ＞

計算機の中で表現できる最大の数、最小の数を考えておこう。 ここでは整数について学んだので、整数の場合について考える。 32ビットのコンピュータ (そこらで見かけるPCである) で表現できる、最大の整数は幾つか。(2の補数なので、先頭の1ビットは符号。だから数を表せるのは31ビット。 とすると…)

負の数についても、同じ議論である。 (但し、負のほうが1つ余分に表現できたりする。2の補数の原理をよく考えよ)

ここで、数を計算をするプログラムを作ってみる。「nの階乗」(n！) を計算するプログラムを書き、nを1ずつ増やしながら、20の階乗までを計算してみよ。 何が起こるか？

［8］　固定小数点による小数の表現　　(教科書 5.1.3)

教科書 5.1.3 を読んで理解する。

＜ことば＞

固定小数点表示とは .....

浮動小数点表示とは .....

正規化とは .....

「ケチ表現」とは .....

教科書 5.1.3 にあるように、小数（実数）を2進数で表すのに、固定小数点と浮動小数点の2種類がある。固定小数点は、整数の表現と同じで、小数点の位置だけをずらす。つまり、整数の場合は小数点が右端にある＝右端の桁が1の位になるのだが、固定小数点の値は小数点の位置がどこでもよいことになる。

原理は10進数と同じである。つまり、12345 は右端の桁の 5 が1の位であった（$1\times10^4 + 2\times10^3 + 3\times10^2 + 4\times10^1 + 1\times10^0$ ）のだが、小数点の位置を 3 と 4 の間に持ってくると、123.45 は 3 の桁が1の位になり、4 の桁が小数点以下第1位になる（$1\times10^2 + 2\times10^1 + 3\times10^0 + 4\times10^{-1} + 1\times10^{-2}$）。 2進数でもまったく同じで、101.01 は左から3桁目の 1 が1の位（2^0に対応）になる（$1\times2^2 + 0\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2}$）。

次に、これらの数を10進法に変換してみよう。計算は上記の($101.01_2 = 1\times2^2 + 0\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2}$)の形を、この式どおりに忠実に計算するだけである。つまり、($1\times2^2 + 0\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2}$) = ($1\times4 + 0\times2 + 1\times1 + 0\times0.5 + 1\times0.25$)。この中で、$2^{-1}$=1/2=0.5 であり、$2^{-2}$=1/($2^2$)=1/4=0.25 である。結果は 5.25 になる。

a.　次の固定小数点2進数を、(1×2^2 ＋ 0×2^1 ＋ 1×2^0 ＋ 0×2^{-1} ＋ 1×2^{-2}) のような形に書き直し、更に10進数に変換せよ。

① $01111.11O_2$　　② 0.0001011_2

③ $010010.1O_2$　　④ 0.1000011_2

逆に、10 進数を2進数に変換するときはどうするか？ （小数点のない）整数の場合には、繰返して2で割ることによって、下の（右側の）桁から求めていっただろう。 たとえば、5_{10}を2進数にする場合、

5/2 = 2 あまり 1　⇒ 余りの 1 を一番右側の桁へ、

上の商の 2 を 2 で割って、2/2 = 1 あまり 0 ⇒ 余りの 0 を右から2桁目へ、

最後に残った 1 を右から3桁目へ

入れる。 これと同じことを小数点以下に対しても行うのだが、小数点以下では≪2 で割る≫代わりに≪1/2 で割る≫、つまり 2 を掛ける。たとえば、0.625_{10}を 2 進数に変換すると、

0.625/(1/2) = 0.625×2 = 1.25 = 1 + 0.25　⇒ 小数点より上に出てきた 1 を一番左側（上位）の桁へ、

小数点以下に残った 0.25 を 1/2 で割って、0.25/(1/2) = 0.25×2 = 0.5 = 0 + 0.25　⇒ 小数点より上へ出てきた 0 を2桁目へ

小数点以下に残った 0.5 を 1/2 で割って、0.5/(1/2) = 0.5×2 = 1　⇒ 小数点より上へ出てきた 1 を3桁目へ入れるので、

0.625_{10} ＝0.101_2となる。　検算すると、$0.101_2 = 0.1_2 + 0.001_2$ = 0.5 + 0.125 = 0.625 である。

b.　次の10進数を2進数に変換せよ

① 0. 4375　　② 0. 1015625　　③ 0. 2109375

c.　10進数の 0. 1 （1／10）を、固定小数点の2進数であらわせ

［9］　浮動小数点による小数の表現　（教科書 5. 1. 3）

（浮動小数点小数）（補足説明）

教科書 45 ページの 5.1.3 を参照。

「有効数字」の考え方は、計算機の浮動小数点表示で初めて出てきたわけではなく、アナログの工学の世界では広く使われてきた。たとえば、長さを定規で読取る、重さをばねばかりで測る、などというとき、目盛り上で実際に読めるのは3桁か4桁であろう。それ以上細かく読んでも、おそらくは誤差の範囲であって、意味がない。このとき、14.6cm = 146mm = 146000 μm と書いてしまうと、後の 000 の部分が実際にはたとえば 237 だが読取りの桁数が足りなくて丸められて 000 になっているにもかかわらず、正確に 000 であるように読めてしまう。 これは困るので、「有効数字は 146 までであり後の 000 は桁表示のためだけのものですよ」 ということを表せるように、146×10^3 や 1.46×10^5 と書く（一般に後者が用いられるが、この文脈では前者も用を為す）。 また 10^5 のように書きにくいので、1.46E+5 のように書くことがある。この場合有効数字は3桁で 1.46 である。また別の例では、三原色の赤い光が 700nm だというのが、0.000007 m と書くと有効数字1桁（7 だけ）に見えるが、7.00E−7 m というのなら有効数字は3桁あることが分かる。この考え方を踏まえて、浮動小数点表示がさまざまなアナログ量の計算に使われる。

浮動小数は、有効数字（「仮数部」、mantissa）＋桁取り（「指数部」、exponent）の2つの要素で書かれる。コンピュータ上の2進浮動小数の表示法は標準規格として決められている。その他に符号ビットがある。

単精度浮動小数（float 型に対応） 符号1ビット＋指数部8ビット＋仮数部23ビット

倍精度浮動小数（double 型に対応） 符号1ビット＋指数部11ビット＋仮数部52ビット

符号は、（符号＋絶対値）表現で、0のとき正で1のとき負

指数部は、2^{-xx} を表すため、「げた履かせ」をする（単精度では 128 を加算）

仮数部は、小数点以下のみを書く＝1桁目が小数点以下1桁目になる。教科書 45 ページの正規化を行う。

a. 符号を S、指数部を E、仮数部を M と書くことにする。 次の単精度浮動小数を10進に直せ

例 S = 0, E = 01111111, M = 0000…000 ⇒ $1.0\times2^0=1.0_{10}$ E は 128 げたはかせのため 0、M はケチ表現（46 ページ下から 5 行目付近）の 1 で 1.0

① S = 0, E = 01111110, M = 0000…000

② S = 0, E = 10000000, M = 0000…000

③ S = 0, E = 10000110, M = 0000…000

④ S = 0, E = 01111111, M = 1100…000

b. 次の10進数を単精度浮動小数（S=..., E=..., M=...の表記でよい）に変換せよ。いずれも正規化が必要である。

① 0.75_{10} ② 3.0_{10} ③ 0.1_{10}

（おまけの問題）

単精度浮動小数では、①有効数字は10進で数えるとおよそ何桁か（たとえば9桁とか12桁とか）、 ②指数部はおよそ10の何乗まで記述できるか。 倍精度浮動小数ではどうか？

（補足説明） たとえば、上記の0.1_{10}の場合、教科書 46 ページにあるように循環小数となるので、仮数部 23 ビットで表示を打ち切ると、「丸め誤差」が出る。これはここで考えた有効数字の桁数にちょうど相当するはずである。誤差の細かいことについては、たとえば http://docs.sun.com/source/806-4847/ncg_goldberg.htm 参照。

≪解答≫

[1]

a. 1×8＋1×4＋0×2＋1＝13 1×32＋1×4＋1×2＝38

1×64＋1×32＋1×8＋1×4＋1＝109 1×128＋1×64＋1×32＋1×16＝240

b. ① 4, ② 8, ③ 16, ④ 32, ⑤ 64, ⑥ 128, ⑦ 256, ⑧ 512, ⑨ 1024
⑩ 4096, ⑪ 32768, ⑫ 65536

[2]

a. ① 11_2 ② 110_2

b. ① 111_2 ② 1111_2

c. ① 0000 0000 0000 0111 ② 0000 0000 0000 1111 ③ 0000 0000 0001 1110 ④ 0000 0000 1111 1110
⑤ 0000 0000 0011 1100 ⑥ 0000 0010 1000 0101

[3]

a. A(=10)、 B(=11)、 C(=12)、 D(=13)、 E(=14)、 F(=15) 2560 176

b. ① 13 ② 90 ③ 13345 ④ 255 ⑤ 4095 ⑥ 65535 ⑦ 65280 ⑧ 60776
④は 0100_{16} ＝(256)$_{10}$ その1つ手前だから、$00FF_{16}$ ⇒ (255)$_{10}$
⑤は 1000_{16} ＝(4096)$_{10}$ なので、その1つ手前だから、 $0FFF_{16}$ ⇒ (4095)$_{10}$
⑥は 10000_{16} ＝(65536)$_{10}$ なので、その1つ手前だから、 $FFFF_{16}$ ⇒ (65535)$_{10}$

c. ① 080B ② 0FFF ③ ACE2 ④ FFFF

d. ① 1010 1011 1100 1101 ② 0001 1111 0101 1011 ③ 1110 1101 0110 1000 ④ 1100 0011 0001 1111

e. $abcdefgh_2 = a\times2^7+b\times2^6+c\times2^5+d\times2^4+e\times2^3+f\times2^2+g\times2^1+h\times2^0$
$= (a\times2^3+b\times2^2+c\times2^1+d\times2^0)\times2^4 + (e\times2^3+f\times2^2+g\times2^1+h\times2^0)$
であるから、 $(abcd)\times16^1 + (efgh)\times16^0$ となる。これは16進数の桁の取り方になっているから、abcd を上の桁、efgh を下の桁として読めば、16進数に読める。

f. ① 1010 1101 1100 1011 ② 0110 1011 0010 1111 ③ 0001 0010 0011 0100 ④ 1011 1100 0001 1010
⑤ 3C0B ⑥ F3C0 ⑦ 000A ⑧ ABCD

g. ① $080B_{16}=8\times16^2+11\times16^0=2048+11 = 2059$
② $0FFF_{16}=15\times16^2+15\times16^1+15\times16^0 = 4095$
③ $ACE2_{16}=10\times16^3+12\times16^2+14\times16^1+2\times16^0 = 44258$
④ $FFFF_{16}=15\times16^3+15\times16^2+15^1+15\times16^0 = 65535$

h. ① 111 ② 1010 ③ 1101 ④ 1111 ⑤ 1 0000 ⑥ 1 1010 ⑦ 101 0000 ⑧ 100 1111
⑨ 1000 0000 ⑩ 111 1111 ⑪ 1 0000 0000 ⑫ 1111 11111 ⑬ 11 1111 1111 ⑭ 1111 1111 1111
⑮ 1111 1111 1111 1111 ⑯ 1 0000 0000 0000 0000

[4]

a. ① 1000 0110 右端の桁を足すと 1+1=10 なので 0 で 1 繰上り。下から2桁目を足すと、0+0 と繰上り 1 を足して 1 で繰上りなし。下から3桁目は 1+0=1 で繰上りなし。以下繰り返せばよい。
② 0100 0011 ③ 1110 0111 ④ 1 0111 1101 ⑤ 0111 0000 0001 0000 ⑥ 1 0110 1100 0011 1000

b. ① 9100 右端の桁を足すと $7+9=16_{10}=10_{16}$ なので 0 で繰上り 1。下から2桁目を足すと、6+9+繰上り $1=16_{10}=10_{16}$ なので 0 で繰上り 1。下から3桁目を足すと、8+8+繰上り $1=11_{16}$ なので 1 で繰上り 1。4桁目は 3+5+繰上り 1=9 で繰上り 0。
② 5544 ③ CF02 ④ 1391B ⑤ AD1D ⑥ F025 ⑦ 4000 ⑧ 8424 ⑨ 19897

[5]

a. ① $+7_{10}=0000\ 0000\ 0000\ 0111_2$ 各桁のビットを反転すると 1111 1111 1111 1000 1を足すと 1111 1111 1111 1001

② 1111 1111 1111 0001　③ 1111 1111 0000 0002　④ 1111 1100 0000 0000
⑤ 1111 1111 1111 1111　⑥ 1111 1111 1111 1110

b. ① $+7_{10}$=0000 0000 0000 0000　0000 0000 0000 0111_2
各桁のビットを反転すると 1111 1111 1111 1111　1111 1111 1111 1000
これに1を足すと 11111 1111 1111 1111　1111 1111 1111 1001
② 1111 1111 1111 1111　1111 1111 1111 0001　③ 1111 1111 1111 1111　1111 1111 0000 0002
④ 1111 1111 1111 1111　1111 1100 0000 0000　⑤ 1111（中略）1111
⑥ 1111 1111 1111 1111　1111 1111 1111 1110

c. ① 1000 0000 0000 0101（符号+絶対値 101）　② 1111 1111 1111 1010（逆転）　③ 1111 1111 1111 1011

d. −5 については6－3の結果を見るといずれの表現でも左端の符号ビットは 1 になっている。
また、+5 についてはどの表現を使っても、0000 0000 0000 0101 である。だから左端の符号ビットは 0 になっている。
つまり左端のビットが 1 ならば負数、0 ならば正数となっており、左端ビットは「負であること」を表す符号ビットである。

e. ① 符号（左端）ビットが 0 なので正数。ならばそのまま2進⇒（16進⇒）10進変換すればよい。16+10=26。
② 符号（左端）ビットが 1 なので負数。この場合、2進表現のまま、2の補数として符号を反転してみる。つまり
ビット反転し1を加える。1111 1111 1111 1010 ⇒ 0000 0000 0000 0101 ⇒ 0000 0000 0000 0110。
これを2進⇒10進変換すると 6_{10} だが、最初に符号反転しているので、それを戻すため「－」（マイナス）を付けて
-6_{10} が答。
③ ①と同じ。そのまま変換して、256+160=416。
④ ②と同じ。一旦符号を反転して 0000 0000 0101 0110 を得る。10進に変換して 80+6=86。これにマイナスを付けて
−86。

f. ① 13　② 90　③ 13345　④ 255　⑤ 4095　⑥ −1　⑦ −256　⑧ −4760

［6］

a. 1の補数　① 0000 0000 0000 0000　② 1111 1111 1111 1111
本来+0 と−0 は同じ数だが、1の補数では +0=0000000000000000 と −0=1111111111111111 の2つの表現が出来てしまう。

b. 2の補数　① 0000 0000 0000 0000　② 0000 0000 0000 0000　で同じ表現になる。

［7］

a. 右図の通り。
−3 を8ビットの2の補数で表すと 1111 1101。 これを8ビットの符号無しとして解釈すると 253_{10} になる。 次に、$253_{10}+5_{10}$ を求める（桁数制限無し）と、258 ＝ 2+256 （1 0000 0010_2）。

```
  0000 0101
+) 1111 1011
------------
1 0000 0000
```

b. ① 0111 1110＋1111 0101 ＝ 1 0111 0011　② 0100 1010＋1111 1101＝1 0100 0111
③ 0111 0111＋1111 0001＝1 0110 1000　④ 0111 1111＋1111 1101＝1 0111 1100

＜おまけ＞の答
プログラムは自分で考えてみよ。 for 文を使って1から100までの和を求めるようなプログラムは、書いたことがあるだろう。それを、足し算の代りに掛け算にし、さらに積の初期値を1にすればよい。（初期値を0のままにすると、0に何を掛けても0だから、空しく終わる。）　⇒ 是非自分の手を動かして、体感してみて欲しい
結果を書き並べてみると、どうだろうか？
1　1
2　2

3　6
4　24
5　120
6　720
7　5040
8　40320
9　362880
10　3628800
11　39916800
12　479001600
13　1932053504
14　1278945280
15　2004310016
16　2004189184
17　-288522240

13 あたりから結果がおかしいことが分かるだろうか？　12 の結果に 13 を掛けても、13 の結果（6 227 020 800）にならない。それどころか 15 と 16 では殆んど増えていないし、17 に至ってはマイナスになっている。
ちなみに、2進31桁の最大数　$2^{31}-1$　は 2 147 483 647 である。

［8］
a.　①　$1\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0 + 1\times2^{-1} + 1\times2^{-2} = 8+4+2+1+0{,}5+0{,}25 = 15.75$
　②　0.00859375　③　18.5　④　0.5234375
b.　①　0.0111　　計算は $0.4375\times2=0.875$　$0.875\times2=1.75$　$0.75\times2=1.5$　$0.5\times2=1$
　②　0.0001101　③　0.0011011
c.　0.0(0011)　カッコ内が循環する　0.0001100110011...
　10 進の切りの良い 0.1 が 2 進で循環小数になることは、たとえば商取引での計算で必ず端数を四捨五入して丸める（つまり常に誤差が出る）ことになるので、問題になることがある。一般には四捨五入すると（正確にはもう少しややこしいが）損得が平均するのだが、切捨にして（数字上は見えない）その端数を自分の口座に集めるようにしたプログラマが捕まったことがある。
［9］
a.　①　$1.0\times2^{-1} = 0.5$　　M はケチ表現で 1.0、E は 128 ゲタはかせで-1 だから。
　②　$1.0\times2^1 = 2.0$　③　$1.0\times2^7 = 128$　④　$1.11_2\times2^0 = 1.75$
b　①　$0.75 = 0.11_2 = 1.1\times2^{-1}$　なので　S=0. E=01111110, M=1000...000
　②　$3 = 11_2 = 1.1\times2^1$　なので　S=0, E=10000000, M=1000...000
　③　$0.1 = 0.000110011..._2 = 1.10011...\times2^{-4}$　なので　S=0, E=01111011, M=10011...

（おまけの問題）
単精度浮動小数は、仮数部＝有効数字を表す部分が2進23ビットであるから、$2^{23}\fallingdotseq10^7$ である。だから、有効数字は10進で約7桁ということになる。
同様に倍精度浮動小数は、仮数部が52ビットであるから、$2^{52}\fallingdotseq4\times10^{15}$である。この場合有効数字は10進で15桁半ということになる。C言語や Fortran のプログラムで倍精度浮動小数を使った計算を行った後、20桁も印刷（表示）している学生がいるが、15桁以降は意味がないということを理解してほしい。

≪基本情報処理技術者試験問題から関連問題≫

1） 2進数の 1.1011 と 1.1101 を加算した結果を 10 進数で表したものはどれか。（基本 14 春 01）

ア 3.1　　イ 3.375　　ウ 3.5　　エ 3.8

2） 10 進数の 0.6875 を2進数で表したものはどれか。（基本 15 春 01）

ア 0.1001　　イ 0.1011　　ウ 0.1101　　エ 0.1111

3） 2進の浮動小数点表示で誤差を含まずに表現できる 10 進数はどれか。(基本 15 秋 01)

ア 0.2　　イ 0.3　　ウ 0.4　　エ 0.5

4） ゼロでない整数の 10 進表示のけた数 D と2進表示のけた数 B との関係を表す式はどれか。（基本 15 秋 02）

ア $D \fallingdotseq 2 \log_{10} B$　　イ $D \fallingdotseq 10 \log_{2} B$

ウ $D \fallingdotseq B \log_{2} 10$　　エ $D \fallingdotseq B \log_{10} 2$

5） 10 進数の演算式 7 ÷ 32 の結果を2進数で表したものはどれか。(基本 16 春 01)

ア 0.001011　　イ 0.001101　　ウ 0.00111　　エ 0.0111

6） 次の式は，何進法で成立するか。(基本 16 春 02)

1015 ÷ 5 ＝ 131 （余り 0 ）

ア 6　　イ 7　　ウ 8　　エ 9

7） ある自然数 x を2進数で表現すると，1 と 0 が交互に並んだ 2 n けたの2進数 1010…10 となった。このとき， x に関して成立する式はどれか。（基本 17 春 02）

ア $x + (x/2) = 2^{2n}$

イ $x + (x/2) = 2^{2n} - 1$

ウ $x + (x/2) = 2^{2n} + 1$

エ $x + (x/2) = 2^{2n+1} - 1$

8） 8ビットで表される符号なし2進数 x が 16 の倍数であるかどうかを調べる方法として，適切なものはどれか。(基本 18 秋 03)

ア x と2進数 00001111 のビットごとの論理積をとった結果が0である。

イ x と2進数 00001111 のビットごとの論理和をとった結果が0である。

ウ x と2進数 11110000 のビットごとの論理積をとった結果が0である。

エ x と2進数 11110000 のビットごとの論理和をとった結果が0である。

9） 1バイトのデータで0のビット数と1のビット数が等しいもののうち，符号なしの2進整数として見たときに最大になるものを，10進整数として表したものはどれか。(基本 18 秋 01)

ア 120　　イ 127　　ウ 170　　エ 240

10） 4ビットの2進数 1010 の1の補数と2の補数の組合せはどれか。（基本 14 春 03）

	1の補数	2の補数
ア	0101	0110
イ	0101	1001
ウ	1010	0110
エ	1010	1001

11） 負の整数を表現する代表的な方法として，次の3種類がある。(基本 19 秋 03)

a 1の補数による表現

b 2の補数による表現

c 絶対値に符号を付けた表現（左端ビットが0の場合は正，1の場合は負）

4ビットのパターン 1101 を a～c の方法で表現したものと解釈したとき，値が小さい順になるように三つの方法を並べたものはど

れか。

ア a , c , b　　イ b , a , c　　ウ b , c , a　　エ c , b , a

12） 負数を2の補数で表すとき，8けたの2進数　n　に対しー n　を求める式はどれか。ここで，＋は加算を表し，OR，XOR は，それぞれビットごとの論理和，排他的論理和を表す。（基本 15 春 03）

ア（ n　OR 10000000）＋ 00000001　　イ（ n　OR 11111110）＋ 11111111

ウ（ n　XOR 10000000）＋ 11111111　　エ（ n　XOR 11111111）＋ 00000001

13）　n　ビットのすべてが 1 である2進数“1111…11”が表す数値又はその数式はどれか。ここで，負数は2の補数で表す。（基本 15 秋 03）

ア　$-(2^{n-1}-1)$　　イ　−1　　ウ　0　　エ　2^n-1

14） 負数を2の補数で表す8ビットの数値がある。この値を 10 進数で表現すると−100 である。この値を符号なしの数値として解釈すると，10 進数で幾らか。（基本 17 春 03）

ア 28　　イ 100　　ウ 156　　エ 228

15） 負数を2の補数で表現する固定小数点表示法において，　n ビットで表現できる整数の範囲はどれか。ここで，小数点の位置は最下位ビットの右とする。（基本 18 春 03）

ア　$-2^n \sim 2^{n-1}$　　イ　$-2^{n-1}-1 \sim 2^{n-1}$

ウ　$-2^{n-1} \sim 2^{n-1}-1$　　エ　$-2^{n-1} \sim 2^{n-1}$

16） 負数を2の補数で表す 16 ビットの符号付き固定小数点数の最小値を表すビット列を，16 進数として表したものはどれか。（基本 18 秋 05）

ア 7FFF　　イ 8000　　ウ 8001　　エ FFFF

17）

負数を2の補数で表すとき，すべてのビットが1であるnビットの2進数 ″1111…11″ が表す数値又はその数式はどれか。（基本 20 春 03）

ア　$-(2^{n-1}-1)$　　イ　−1　　ウ　0　　エ　2^n-1

18） 2の補数で表された負数 10101110 の絶対値はどれか。（基本 20 秋 03）

ア 01010000　　イ 01010001　　ウ 01010010　　エ 01010011

19） 多くのコンピュータが，演算回路を簡単にするために補数を用いている理由はどれか。（基本 17 秋 05）

ア 加算を減算で処理できる。

イ 減算を加算で処理できる。

ウ 乗算を加算の組合せで処理できる。

エ 除算を減算の組合せで処理できる。

20） 負数を2の補数で表す 16 ビットの符号付き固定小数点方式で，絶対値が最大である数値を 16 進数として表したものはどれか。（基本 14 秋 03）

ア 7FFF　　イ 8000　　ウ 8001　　エ FFFF

21） 実数 a を $a = f \times r^e$ と表す浮動小数点表示に関する記述として，適切なものはどれか。（基本 21 秋 02）

ア　f を仮数，e を指数，r を基数という。

イ　f を基数，e を仮数，r を指数という。

ウ　f を基数，e を指数，r を仮数という。

エ　f を指数，e を基数，r を仮数という。

22） 基数変換に関する記述のうち，適切なものはどれか。（基本 20 秋 02）

ア 2進数の有限小数は，10 進数にしても必ず有限小数になる。

イ 8進数の有限小数は，2進数にすると有限小数にならないこともある。

ウ 8進数の有限小数は，10 進数にすると有限小数にならないこともある。

エ 10 進数の有限小数は，8進数にしても必ず有限小数になる。

≪基本情報処理技術者試験の答≫

1）ウ　2）イ　3）エ　4）エ　5）ウ　6）イ　7）イ　8）ア　9）エ　10）ア
11）エ　12）エ　13）イ　14）ウ　15）ウ　16）イ　17）イ　18）ウ　19）イ　20）イ
21）ア　22）ア

第２回補足　（論理演算の復習、正解無し） 下記に、基本情報技術者試験の問題のうち、論理演算に関わるものを抜粋してある。自分でできることを確認しておいて欲しい。

1） 任意のオペランドに対するブール演算 A の結果とブール演算 B の結果が互いに否定の関係にあるとき，A は B の（又は，B は A の）相補演算であるという。　排他的論理和の相補演算はどれか。（基本 14 春 09）

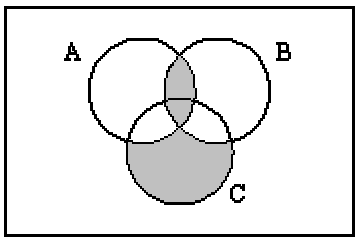
ア 等価演算 

イ 否定論理和 

ウ 論理積 

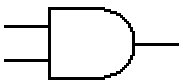
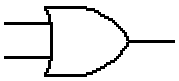
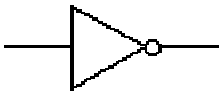
エ 論理和 

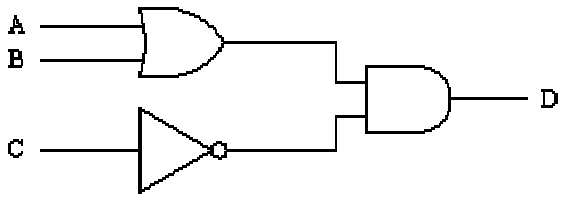
2） 次のベン図の網掛け部分（  ）で表現される集合はどれか。ここで，X ∪ Y は X と Y の和集合，X ∩ Y は X と Y の積集合，$\overline{X}$ は X の補集合を表す。（基本 14 秋 05）



ア $(A \cup B) \cap C$　　イ $(A \cap B) \cup (C \cap \overline{A \cup B})$

ウ $\overline{(\overline{A} \cap \overline{B}) \cap C}$　　エ $\overline{C} \cap (A \cup B)$

3） 図の論理回路と等価な論理式はどれか。ここで，  は AND ゲート，  は OR ゲート，  は NOT ゲートとする。また，・は論理積，＋は論理和，$\overline{X}$ は X の否定を表す。（基本 14 秋 06）



ア （A+B）・C ＝ D　　イ （A+B）・$\overline{C}$ ＝ D

ウ （A・B）＋C ＝ D　　エ （A・B）＋$\overline{C}$ ＝ D

4） 真理値表と等価な論理式はどれか。ここで，・は論理積，＋は論理和，$\overline{A}$ は A の否定を表す。（基本 14 秋 07）

x	y	演算結果
0	0	0
0	1	0

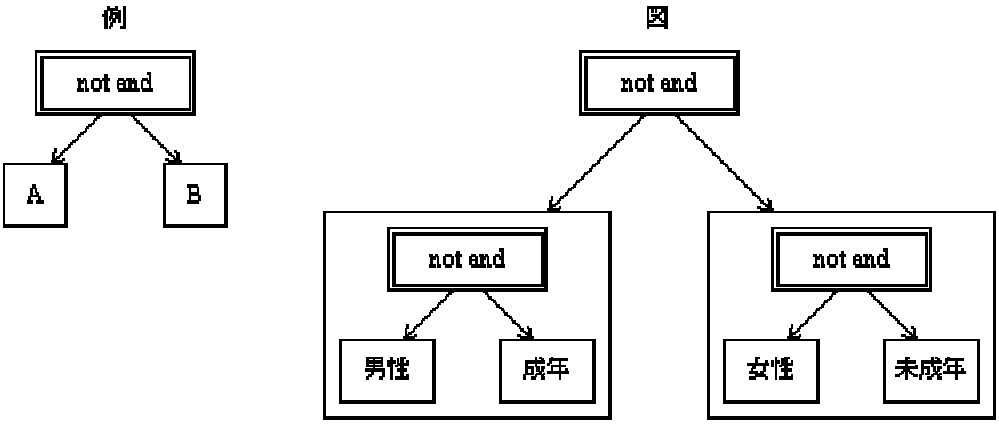
1	0	1
1	1	0

ア $x + \overline{y}$　　イ $\overline{x} + y$　　ウ $x \cdot \overline{y}$　　エ $\overline{x} \cdot y$

5） 論理式 A ∨（$\overline{A}$ ∧ B ）と等価なものはどれか。 ここで， ∧ は論理積， ∨ は論理和， $\overline{X}$ は X の否定を表す。（基本 15 春 07）

ア A ∧ B　　イ A ∨ B　　ウ A ∧ $\overline{B}$　　エ A ∨ $\overline{B}$

6） 論理式 $\overline{A \wedge B}$ を例のとおりに記述するとき，図で記述される論理式が表すものはどれか。（基本 15 春 08）



ア 女性　　イ 成年男性又は未成年女性
ウ 男性　　エ 未成年男性又は成年女性

7） 二つの入力と一つの出力をもつ論理回路で，二つの入力 A ， B がともに 1 のときだけ，出力 X が 0 になるものはどれか。（基本 15 秋 17）

ア AND 回路　　イ NAND 回路　　ウ OR 回路　　エ XOR 回路

8） 1ビットの数 A ， B の和を2ビットで表現したとき，上位ビット C と下位ビット S を表す論理式の組合せはどれか。ここで，“・”は論理積，“＋”は論理和，$\overline{X}$ は X の否定を表す。（基本 16 春 07）

A	B	A と B の和	
		C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

	C	S
ア	A ・ B	$(A \cdot \overline{B}) + (\overline{A} \cdot B)$

イ	A ・ B	$(A+\overline{B})\cdot(\overline{A}+B)$
ウ	A ＋ B	$(A\cdot\overline{B})+(\overline{A}\cdot B)$
エ	A ＋ B	$(A+\overline{B})\cdot(\overline{A}+B)$

9） 8ビットのデータの下位2ビットを変化させずに，上位6ビットのすべてを反転させる論理演算はどれか。（基本 16 秋 08）

ア 16 進数 03 と排他的論理和をとる。

イ 16 進数 03 と論理和をとる。

ウ 16 進数 FC と排他的論理和をとる。

エ 16 進数 FC と論理和をとる。

10） 16 ビットの符号なし固定小数点の2進数 n を， 16 進数の各けたに分けて下位のけたから順にスタックに格納するために，次の手順を 4 回繰り返す。a，b に入る適切な語句の組合せはどれか。ここで，$xxxx_{16}$ は 16 進数 xxxx を表す。（基本 16 秋 02）

〔手順〕 (1)［ a ］を x に代入する。

(2) x をスタックにプッシュする。

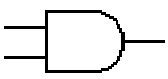
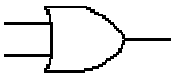
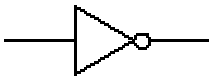
(3) n を［ b ］論理シフトする。

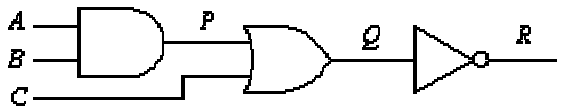
	a	b
ア	n AND $000F_{16}$	左に 4 ビット
イ	n AND $000F_{16}$	右に 4 ビット
ウ	n AND $FFF0_{16}$	左に 4 ビット
エ	n AND $FFF0_{16}$	右に 4 ビット

11）X ・ Y ・ Z ＋ $\overline{X}$ ・ Y ・ Z と等価な論理式はどれか。ここで，“・”は論理積，“＋”は論理和，$\overline{X}$ は X の否定を表す。（基本 16 秋 09）

ア X ・ Y ・ Z　　イ $\overline{X}$ ・（Y ＋ Z）　　ウ Y ・ Z　　エ Y ＋ Z

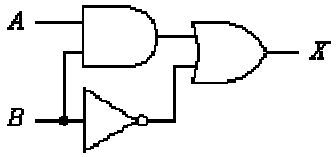
12） 図の論理回路において， A ＝1， B ＝0， C ＝1 のとき， P ， Q ， R の値の適切な組合せはどれか。ここで，

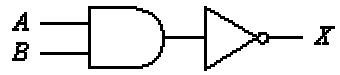
 は AND 回路，  は OR 回路，  は NOT 回路を表す。（基本 16 秋 17）

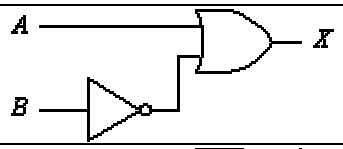


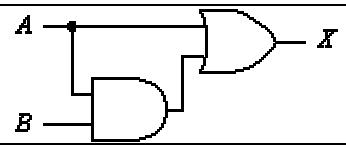
	P	Q	R
ア	0	1	0
イ	0	1	1
ウ	1	0	1
エ	1	0	0

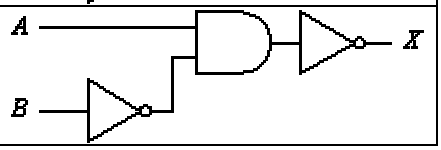
13） 図の論理回路と同じ出力が得られる論理回路はどれか。ここで，  は論理積（AND），  は論理和（OR），  は否定（NOT）を表す。（基本 17 春 17）



ア 

イ 

ウ 

エ 

14） 次に示す手順は，列中の少なくとも一つは1であるビット列が与えられたとき，最も右にある1を残し，ほかのビットをすべて0にするアルゴリズムである。例えば，00101000 が与えられたとき，00001000 が求まる。[a]に入る論理演算はどれか。（基本 18 秋 08）

手順1 与えられたビット列 A を符号なしの2進数と見なし，A から1を引き，結果を B とする。
手順2 A と B の排他的論理和（ XOR ）を求め，結果を C とする。
手順3 A と C の [a]を求め，結果を A とする。

ア 排他的論理和（ XOR ） イ 否定論理積（ NAND ） ウ 論理積（ AND ） エ 論理和（ OR ）

15） ビット数が等しい任意のビット列 a と b に対して，等式 a ＝ b と同じことを表すものはどれか。ここで，AND，OR，XOR はそれぞれ，ビットごとの論理積，論理和，排他的論理和を表す。（基本 14 春 08）

ア a AND b ＝ 00…0 イ a OR b ＝ 11…1
ウ a XOR b ＝ 00…0 エ a XOR b ＝ 11…1

16） 最上位をパリティビットとする8ビット符号において，パリティビット以外の下位7ビットを得るためのビット演算はどれか。（基本 15 秋 06）（基本 18 春 06）

ア 16 進数 0F との AND をとる。
イ 16 進数 0F との OR をとる。
ウ 16 進数 7F との AND をとる。
エ 16 進数 FF との XOR（排他的論理和）をとる。

17） 8ビットのビット列の下位4ビットが変化しない操作はどれか。（基本 17 春 09）

ア 16 進表記 0F のビット列との論理積をとる。
イ 16 進表記 0F のビット列との論理和をとる。
ウ 16 進表記 0F のビット列との排他的論理和をとる。
エ 16 進表記 0F のビット列との否定論理積をとる。

18） X と Y の否定論理積 X NAND Y は， NOT (X AND Y)として定義される。 X OR Y を NAND だけを使って表した論

理式はどれか。（基本 17 秋 09）

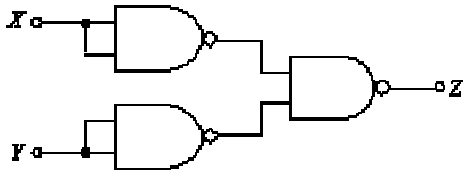
ア （(X NAND Y) NAND X) NAND Y

イ (X NAND X) NAND (Y NAND Y)

ウ (X NAND Y) NAND (X NAND Y)

エ X NAND (Y NAND (X NAND Y))

19） NAND 回路による次の組合せ回路の出力 Z を表す式はどれか。ここで，  は NAND 回路，・は論理積，＋は論理和，$\overline{X}$ は X の否定を表す。（基本 18 春 16）



ア X・Y　　イ X＋Y　　ウ $\overline{X+Y}$　　エ $\overline{X \cdot Y}$

20） 論理式 $Z = X \cdot \overline{Y} + \overline{X} \cdot Y$ の真理値表はどれか。ここで，・は論理積，＋は論理和，$\overline{X}$ は x の否定を表す。（基本 15 秋 08）

ア

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

イ

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

ウ

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

エ

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	1

21） 次の真理値表で，変数 X ， Y ， Z に対する関数 F を表す式はどれか。ここで，“・”は論理積，“＋”は論理和，$\overline{A}$ は A の否定を表す。（基本 18 秋 09）

X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

ア $X \cdot Y + Y \cdot \overline{Z}$　　イ $X \cdot Y \cdot \overline{Z} + Y$　　ウ $\overline{X} \cdot \overline{Y} \cdot Z + X \cdot Y + Y \cdot \overline{Z}$　　エ $\overline{X} \cdot \overline{Y} \cdot Z + X \cdot \overline{Y} + \overline{Y} \cdot \overline{Z}$

22） 次の真理値表の演算結果を表す論理式はどれか。ここで，＋は論理和，・は論理積を表す。（基本 20 秋 10）

x	y	z	演算結果
0	0	0	0
0	0	1	0
0	1	0	0

0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

ア （ x ・ y ）＋ z　　イ （ x ＋ y ）・ z　　ウ x ・（ y ＋ z ）　　エ x ＋（ y ・ z ）

23） 次の状態遷移表をもつシステムの状態が S1 であるときに，入力信号（ t1, t2, t3, t4, t1, t2, t3, t4 ）を順次入力したとき，最後の状態はどれか。ここで，空欄は状態が変化しないことを表す。（基本 14 秋 10）（基本 19 秋 10）

状態 / 信号	S1	S2	S3	S4
t1		S3		
t2	S3		S2	
t3			S4	S1
t4		S1		S2

ア S1　　イ S2　　ウ S3　　エ S4

24） 次の表は，入力文字列を検査するための状態遷移表である。この検査では，初期状態を a とし，文字列の入力中に状態が e になれば不合格とする。

解答群で示される文字列のうち，この検査で不合格となるものはどれか。ここで，解答群中の△は空白を表す。（基本 16 秋 11）（基本 18 春 09）

		入力文字				
		空白	数字	符号	小数点	その他
現在の状態	a	a	b	c	d	e
	b	a	b	e	d	e
	c	e	b	e	d	e
	d	a	e	e	e	e

ア ＋0010　　イ －1　　ウ 12.2　　エ 9.△

25） 状態遷移図を用いて設計を行うことが最も適しているシステムはどれか。（基本 15 春 48）（基本 20 秋 40）

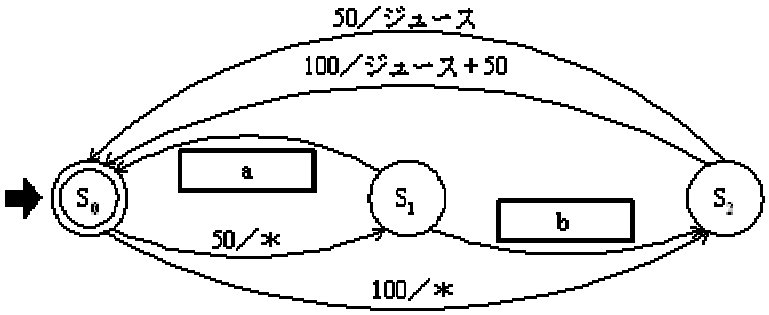
ア 月末及び決算時の棚卸資産を集計処理する在庫棚卸システム

イ システム資源の稼働状態を計測し，レポートとして出力するシステム資源稼働状態計測システム

ウ 水道の検針データから料金を計算する水道料金計算システム

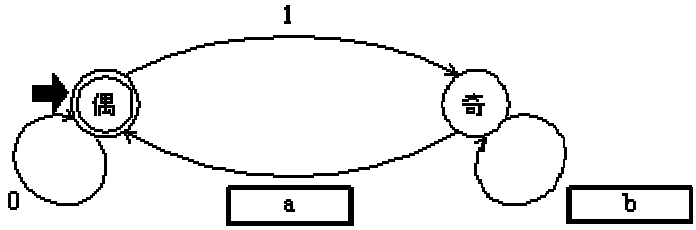
エ 設置したセンサの情報から，温室内の環境を最適に保つ温室制御システム

26） 図は，150 円のジュースを販売する自動販売機の状態遷移において，状態を “Si”，遷移条件を“X ／ Y ＋ Z ”で表したものである。“S0”を初期状態とすると，図中の a, b に入れるべき字句の適切な組合せはどれか。ここで，X は入力を示し，使用可能な硬貨は 50 円と 100 円だけであり，一度に1枚だけ投入できる。Y は出力を示し，＊は何も出力されないことを表す。また，Z は X と Y による付帯条件“釣銭”を表し，釣銭がない場合は記述しない。例えば，“100 ／ジュース＋ 50 ”は，100 円硬貨を投入するとジュースが出て，釣銭が 50 円であることを表す。 （基本 15 春 10）



	a	b
ア	100 ／＊	50 ／＊
イ	100 ／ 50	50／ジュース
ウ	100 ／ジュース	50 ／＊
エ	100 ／ジュース	50 ／ジュース

27） 図は1の数が偶数個のビット列を受理するオートマトンの状態遷移図であり，“偶”と書かれた二重丸が受理状態を表す。a, b の正しい組合せはどれか。（基本 17 春 11）



	a	b
ア	0	0
イ	0	1
ウ	1	0
エ	1	1

１）ア　２）イ　３）イ　４）ウ　５）イ　６）イ　７）イ　８）ア　９）ウ　１０）イ
１１）ウ　１２）ア　１３）イ　１４）ウ　１５）ウ　１６）ウ　１７）ア　１８）イ　１９）イ　２０）イ
２１）ウ　２２）ウ　２３）ア　２４）ウ　２５）エ　２６）ウ　２７）ウ