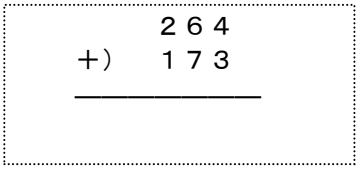
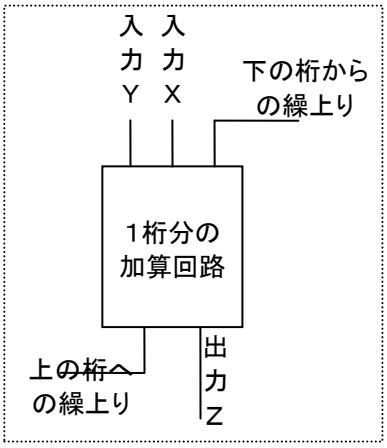
第３回　演算の仕組　予習資料

第3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］（整数）加算回路　　（教科書 5.2.1）
教科書 5.2.1 を読んで理解しておくこと。



16ビット（や32ビット）の2進数同士を足し合わせる回路を作る。
（考え方） 我々が10進数で足し算をするとき、「下の桁から」「1桁ずつ」足す。右図の例では、まず1の桁の4＋3を足して7を得る。この1桁の足し算は表の形で知っている。
次に10の桁の6＋7を足して、13を得る。10を超えるので上の桁へ1繰り上がる。この桁には3を残す。
100の桁の2＋1を足して3を得るが、下の桁からの繰上り1をもらって3＋1で4になる。
つまり、各桁を足すので、1桁分の足し算回路を作って、それを桁数の分だけ並べればよさそうだ。16ビット（＝16桁）なら、16個並べればよかろう。 但し、それぞれの桁の足し算回路の入力は、足したい2つの数だけではなくて、下の桁からの繰上りも足さなければならない。だから入力は3本になる。 それぞれの桁の出力も、その桁に残す値だけではなくて、もし繰上りが出ればその繰り上がりの値を上の桁に渡してやらなければならない。



a.　1桁分の加算回路（フルアダー）の入力と出力は何か
　入力（　　）（　　）（　　）
　出力（　　）（　　）

b.　1桁分の加算回路（フルアダー）のそれぞれの出力信号を、入力の値から作るためには、どのような入力と出力の値の関係（要するに関数の関係）があればよいのか。たとえば、入力X＝1、入力Y＝0、下の桁からの繰上り＝1の場合の、出力Zの値はいくつであるべきか、上の桁への繰上り出力はいくつであるべきか？
　出力Z（　　）　上への繰上り出力（　　）　なぜか（　　）

c.　上で考えた、1桁の加算回路（フルアダー）の入出力の（関数）関係を、真理値表の形に書け

d.　上で考えた1桁の加算回路（フルアダー）を用いて、8桁の整数（符号なし）の加算回路を作りたい。1桁加算回路を箱（ブラックボックス）と考えて、それを8個並べて作ることができるだろう。 1桁加算回路をどのように接続したらよいか。接続図を書け。

e.　1桁の加算回路（フルアダー）を単純に接続しただけの回路は、「リップルキャリー」式と呼ばれる。この回路は加算に時間がかかると言われる。 その理由を説明せよ。

f.　上記のリップルキャリー方式における遅延を、改善（短縮）するための対策として、どのようなものが考えられるか。ネット等で調べてみよ。

［２］　引き算　（教科書 5.2.1）

引き算A－Bは、引く数Bの「2の補数」を取ることによって（－B）を作り、それをAと加えることによって、A＋（－B）を計算すれば良い。 従って、Bの「2の補数」を作る回路を作ればよい。

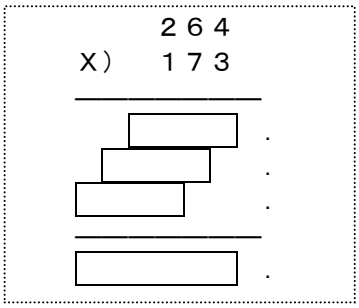
a.　2の補数を作るために、入力Bのビットを反転したい。教科書 p48 の図 5.8 の中で、入力Bのビットを反転するための回路はどの部分か？

（注） 選択信号 S_0、S_1 に対して、S_0＝1、S_1＝0を与えると入力Bを反転せず普通に足し算するが、S_0＝0、S_1＝1を与えると入力Bを反転し、2の補数を作るときに使うようになっている。

b.　入力Bのビットを反転した後、1を加える。教科書 p48 の図 5.8 の中で、1を加える動作は、どの部分か？

［３］　掛け算　（教科書 5.2.2）

教科書 5.2.2 を読んで理解する。授業では負の数の扱いとブース法については触れないこととし、正の整数同士の掛け算を理解すればよいこととする。



a.　掛け算の手順を考える。右図の10進数の掛け算 264×173 を筆算で行うとするとき

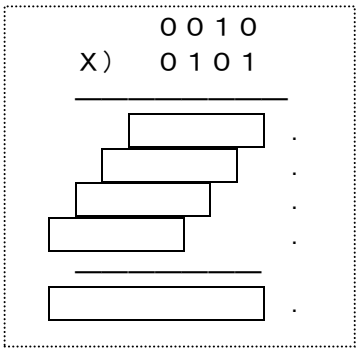
① それぞれの段に入る数値は何か

64×7の結果を第2桁目にあわせてずらして書く。 同様に、第3桁“1”（173の左端の”1”） を掛ける時に、264×1の結果を第3桁目にあわせてずらして書く。これがどういう意味かを式で説明せよ。

b.　では、これと同じ原理を、2進数 0010×0101 で行う（右図に従う）とどうなるか。

① それぞれの段に入る数値は何か

② この手順の中で、2段目の第3桁“0”（0101の左から3つ目の”1”） を掛ける時に、0010×1の結果を第3桁目にあわせてずらして書く。 これがどういう意味かを式で説明せよ。



c.　上の考察から、桁ずらしの操作ができれば、2進数の掛け算は、筆算の手順どおりに実現できる。2進数での左への1桁分の桁ずらしは、全体を2倍する効果があるが、それはシフト命令（シフト回路）で行うことができる。教科書の 5.2.2 節を参考にしながら、2進数の掛け算の計算手順を書いてみよ。 入力をX、Y、結果をZとし、Yのそれぞれの桁を表すY_iの添字 i は右側＝下位の桁から左へ向かって0, 1, 2, 3としよう。 下記のプログラムで、［　　］の部分に入る正しい組合わせはどれか

```
class mult1 {
  public static void main(String args[]) {
    int x = 2;
    int y = 5;
    int z = 0;
    for (int i=0; i<4; i++) {
      // y の i 桁目を取り出す
      [　　ア　　];

      // yi が 1 なら x をそのまま、0 なら 0 を、ABCD や EFGH に作る
      // 但し、EFGH なら左へ 1 ビット、IJKL なら 2 ビット、MNOP なら 3 ビットシフトして足し合わせる
      if (yi == 1) {
        [　　イ　　];
```

```
        } else {
            [ウ];
        }
      }
      System.out.println(z);
    }
```

① ア： z = z + (x << i)　　イ： z = z + 0　　ウ： int yi = (y & (1 << i)) >> i

② ア： int yi = (y & (1 << i)) >> i　　イ： z = z + (x << i)　　ウ： z = z + 0

③ ア： int yi = (y & (1 << i)) >> i　　イ： z = z + 0　　ウ： z = z + (x << i)

④ ア： int yi = (y & (1 << i)) >> i　　イ： z = z + x　　ウ： z = z + 0

但し、演算子「&」はビット AND （p&q は p の各ビットと q の各ビットを AND する。10111&01101⇒00101）、演算子「<<」は左シフト（101<<2⇒10100）、演算子「>>」は（算術）右シフトを表す。

d. 上記のようなプログラムを Java 言語で作成し、実際に PC 上で動かしてみよ。 プログラムは改良の余地がたくさんあるので、自分の思うように改良し、うまく動くか確かめよ。 X と y にいろいろな値を入れてみて、結果を印刷せよ。
また、任意の桁数を扱えるように改良してみよ。

（補足説明: 論理シフトと算術シフト）
コンピュータのシフト命令（教科書 27 ページ、49 ページ、ただし命令については次回から学ぶ）について、確認しておく。
シフト演算は、2進データを左又は右に桁をずらす処理である。たとえば 0010 を左に1ビットシフトすれば、0100 になる。
上の問題で見たように、1ビット左シフトするということは、2倍することに当る。たとえば 0010=2 を1ビット左シフトすると 0100=4 になる。逆に、1ビット右シフトするということは、2で割る（2^{-1}倍する）ことに当る。たとえば 0100=4 を1ビット右シフトすると 0010=2 になる。
COMET Ⅱ （本授業の5回目以降で例として使う仮想的なコンピュータ） では、1回のシフト命令では1ビットだけシフトする。3ビットシフトするためには3回シフトを行う必要がある。最近の汎用 CPU（例えば Pentium/Core2 など）では1回の命令で任意ビット数（たとえば1～32ビットまでの任意の量）だけシフトできるものが主流である。
シフトには、論理シフトと算術シフトの2種類がある。論理シフトでは、シフトの結果はみ出したビットは捨てられ、新しく空いたビットには0が詰められる。算術シフトでは、はみ出したビットは同様に捨てられるが、新しく空いたビットには、右シフトに限り、状況によって0が詰められたり1が詰められたりする。具体的には、右算術シフトで新しく空いたビット（＝左端のビットになる）には、シフト前の左端のビットの値を詰める。もしシフト前の左端のビットが1であれば、算術右シフトにより左端ビットに1が詰められる。たとえば、1010 を1ビット右算術シフトすると、（左端が 1 であるから、左から 1 を詰めて）1101 になる。もしシフトする前の左端のビットが 0 であれば、算術右シフト後の左端空きビットに 0 を詰める。たとえば、0111 を1ビット右算術シフトすると、0011 になる。（算術シフトではなくて論理右シフトの場合は 0101 のように、左端ビットの値に関わらず、左から 0 を詰める）。

e. 5ビットの整数で（桁数が半端だが簡単化のため我慢せよ）、次の2進数を指定どおりシフトせよ
① 00110 を2ビット「論理」右シフトせよ　② 00110 を2ビット「算術」右シフトせよ
③ 10110 を2ビット「論理」右シフトせよ　④ 10110 を2ビット「算術」右シフトせよ

算術右シフトが必要な理由を考えてみよう。整数が2の補数で表現されているとすると、実は左端のビットは符号を表している。たとえば、1010_2 を2の補数表現だと思うと、-6_{10} に当る。 これを単純に1ビット右（論理）シフトすると、$0101_2=+5_{10}$ になり、負の数が正になってしまう。これに比べて、1ビット右算術シフトすると、$1101_2=-3_{10}$ になる。つまり、-6 を1ビット右シフトする

と-3 になる。つまり、算術右シフトは、負の数に対して（1ビット右シフト＝2で割る）という意味を正のときと同様に成り立たせる効果がある。 なお、算術右シフトでは符号ビットを左端から空きビットに埋めていくので、これを「符号拡張」（sign extension）と呼ぶ。 また、左シフトについては符号拡張は無い。
なお、符号拡張は、シフトだけでなく、データのビット幅の拡大時にもしなければならないことがある。たとえば、8ビット整数データを16ビットや32ビット整数として読み直す～型変換する～のような場合である。

f. 8ビット整数で2の補数表現をするものとする。 00101001_2 （＝（　　）$_{10}$）に対して次のシフト演算するとどうなるか。また、それぞれの結果を10進数に変換し、理解した状況を説明せよ。

① 00101001_2を3ビット論理左シフト ⇒ （　　）$_2$ ＝ （　　）$_{10}$
状況の説明 （$00101001=41_{10}$を左に3ビットシフトしたため、　　）

② 00101001_2を3ビット論理右シフト ⇒ （　　）$_2$ ＝ （　　）$_{10}$
状況の説明 （$00101001=41_{10}$を右に3ビット論理シフトしたため、　　）

③ 00101001_2を3ビット算術右シフト ⇒ （　　）$_2$ ＝ （　　）$_{10}$
状況の説明 （$00101001=41_{10}$を右に3ビット算術シフトしたため、　　）

g. 8ビット整数で2の補数表現をするものとする。 11010110_2 に対して次のシフト演算するとどうなるか。上記と同様に、結果を10進に変換し、理解した状況を説明せよ

① 11010110_2を3ビット論理左シフト ⇒ （　　）$_2$ ＝ （　　）$_{10}$
状況の説明 （11010110＝（　　）$_{10}$を左に3ビットシフトしたため、　　）

② 11010110_2を3ビット論理右シフト ⇒ （　　）$_2$ ＝ （　　）$_{10}$
状況の説明 （11010110＝（　　）$_{10}$を右に3ビット論理シフトしたため、　　）

③ 11010110_2を3ビット算術右シフト ⇒ （　　）$_2$ ＝ （　　）$_{10}$
状況の説明 （11010110＝（　　）$_{10}$を右に3ビット算術シフトしたため、　　）

（補足説明）
負の数の表し方（絶対値表現 vs 補数表現）と演算の種類は、深く関わっている。足し算と引き算が2の補数表現でかなりうまく行くことは、前に例題で見ている。たとえば 5+(−3)が、−3 の補数表現を正しく取ってあれば、正の数同士の足し算と同じ手順で計算でき、負数であるための特別な計算手順は必要ない。もし絶対値表現された-3（つまり符号−と、正の 3 と同じ表現をされた値 3）を用いると、3 を「引く」ための論理処理（回路）が必要になる。
ところが一方、掛け算・割り算は絶対値表現の方がよい。なぜなら、5×(−3) は−(5×3) として計算されるからである。掛け算の論理処理は、符号を除けておいて絶対値同士を掛け算し、それに符号を付けることで得られる。（教科書 49 ページに出ている Booth の掛け算アルゴリズムはそこに一工夫して、2の補数で掛け算ができるようにしている。）
（補足説明）
上記のように1ビットシフトしては足し込むというやり方は、桁数が増えると時間がかかるようになるので、ハードウェアで高速化するような工夫がされている。興味のある人はウォレス・トリー（Wallace Tree）などを調べてみるとよい。

［文字コード］ （授業外、 教科書 5.1.4）
文字コードについて、教科書 5.1.4 で簡単に触れているが、ここでコメントしておく。
（主としてハードウェアの問題ではなくてソフトウェアの問題に当たる）、次のことについて考えてみよ。

文字も、計算機の内部では2進数で表す。英字・数字を表すのに2進数で何ビット必要か、考えてみる。
3ビットの2進数は 0, 1, 2, … 7(=2^3−1) の8（=2^3）通りのパターンを表すことができる。一般に、Nビットの2進数は2^N通りのパターンを表すことができる。

① アルファベットは 26 文字、それを2進数で表すと(　　)ビット(2(　　)=(　　)パターンが表せる)必要である。
② 数字が 10 文字、それを2進数で表すと(　　)ビット(2(　　)=(　　)パターンが表せる)必要である。
③ 従って、アルファベットと数字を表すには、合計(　　)ビット必要である。
　というのは実は誤りで、
④ アルファベットは 26 文字、数字が 10 文字なので、合計 36 文字、それを2進数で表すと(　　)ビット(2(　　)=(　　)パターンが表せる)必要である
⑤ さらには、英字で大文字と小文字を区別することとし(もう26文字増える)、更に以下の33個の記号(ASCIIコードの場合)を付け加える。 合計何文字か？ それを表すのに2進数で何ビット必要か？
(スペース) ! ″ # $ % & ' ( ) * + , − . / : ;
< = > ? @ [¥] ^ _ ` { | } ~
⑥ 英数字を表すのに広く用いられている「ASCII コード」の場合、英字・数字・記号の他に「制御文字」と呼ばれる文字がある(下記の 33 文字)。 それらすべてを表すのに2進数で何ビット必要か？
　NUL(null 文字)　SOH(ヘッダ開始)　STX(テキスト開始)　ETX(テキスト終了)　EOT(転送終了)　ENQ(照会)
　ACK(受信OK)　BEL(警告)　BS(後退)　HT(水平タブ)　LF(改行)　VT(垂直タブ)　FF(改頁)　CR(復帰)
　SO(シフトアウト)　SI(シフトイン)　DLE(データリンクエスケープ)　DC1(装置制御1)　DC2(装置制御2)
　DC3(装置制御3)　DC4(装置制御4)　NAK(受信失敗)　SYN(同期)　ETB(転送ブロック終了)　CAN(とりけし)
　EM(メディア終了)　SUB(置換)　ESC(エスケープ)　FS(フォーム区切り)　GS(グループ区切り)
　RS(レコード区切り)　US(ユニット区切り)　DEL(削除)

日本語をコンピュータで扱おうとすると、かな文字(ひらがな、カタカナ)と漢字に対して文字コードを対応させる必要があるだろう。ひらがな・カタカナ・漢字の総数は、およそ5万文字(康熙字典)と言われるが、日本語で常用する漢字としておよそ6350文字が定められており、コンピュータ内ではそれを16ビットを用いて表している
[補足1] 常用漢字、JIS 漢字コードを調べてみよ。
[補足2] コンピュータ上でかな・漢字を扱うためのコード体系(どの文字をどの2進数に割当てるか)には、歴史的経緯から、いくつかの体系が併用されており、混乱している。次のものについて、どういうものか、どう違うのか、調べてみよ。
　JIS 漢字コード、 Shift-JIS コード、 EUC コード、 ユニコード(unicode)、 UTF-8
＊もし興味があれば、日本語以外の言語の文字コードについても調べてみると面白いだろう。
＊更に、複数の言語が混在する文書(たとえばホームページとか)を考えてみよ。同一の2進数体系の中で、英字と漢字とハングルとサンスクリットを同時に表示したいとき、何が起こるだろうか？

[情報落ちと桁落ち]　(授業外)
教科書では取り上げていないが、基本情報技術者試験でよく出ている「情報落ち」と「桁落ち」について簡単に触れて置く。
＜情報落ち＞
情報落ちとは、「絶対値の非常に大きな数と小さな数の足し算や引き算を行ったとき，小さい数が計算結果に反映されないために発生する誤差」(基本情報2種 11 年春 10)と言える。例で考えてみる。簡単のため 10 進とし、ある計算機では 10 進 4 桁しか表現できないとするとき、
　a = 1.234 × 10^{+2} = 123.4　　b = 5.678 × 10^{-2} = 0.05678
の場合に、a + b は 123.45678 だが4桁なので 123.4 となってしまい、b は反映されず誤差となる現象が起こる。
＜桁落ち＞
桁落ちとは、「絶対値のほぼ等しい二つの数の絶対値の差を求めたとき，有効けたが減るために発生する誤差」(同上)である。同様に例で考えると、
　a = 1.234　　b = 1.233
とすると、 a−b = 0.001 だが、この結果は有効数字としては1桁しかないことになる。

ウィキペディア　http://ja.wikipedia.org/wiki/誤差　に、1001 の平方根から 999 の平方根を引く計算が載っている。そのまま引き算をすると桁落ちが出るので、式変形して引き算の無い計算式に変換してから求める例が挙げられているので、見ておくとよい。

また、私自身の好きな例としては、

地球にハチマキをする。赤道上の地面に沿って、ロープを一周させる。長さは 40075.017 Km だが、そのロープに 10m 追加する。その緩んだ分を赤道上で同じ高さにすると、その高さはどれだけか？言い換えると、40,075,017m＋10m のロープを一周したときの半径と、地球の半径との差はどれだけか？

計算すると、(40075027/2π) － (40075017/2π) ＝6378138.6416…－6378137.0500＝1.516m となる。 単精度浮動小数を使うと、仮数部が 23 ビットつまり有効数字は約 $2^{23}=10^{7}$ で 10 進 7 桁になるので、6378139－6378137＝2m となって、答の有効数字は1桁となり、思っている値とはかなり違う。もし10進6桁しかなければ、6378140－6378140＝0m という答が返ってくる。

この例題の場合、計算順序を変えることで桁落ちを回避できる。(40075027/2π)－(40075017/2π)＝(40075027－40075017)/2π　＝10/2π＝1.591549m といった精度で計算できる。

≪解答≫

[1]

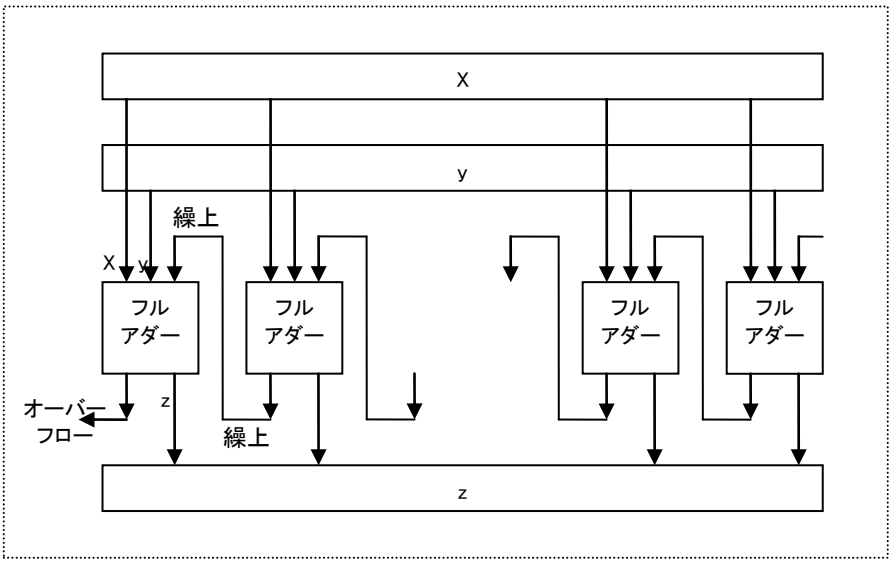
a.　入力 (X), (Y) と (繰上り入力)。 出力は (Z) と (繰上り出力)。

b.　入力X＝1, Y＝0, 繰上り入力＝1だと、入力される1の数が2つなので和は $2_{10}=10_{2}$ となる。この 10_{2} は、1の位の値を出力Zとして、2の位の値を繰上り出力として、出力する。 だから、Z＝0で、上への繰上り＝1。

c.　このようにして、全ての入力パターンについて出力Zと繰上り出力の値を決める。その結果(右の表)が答えになる。　繰り返すが、3本の入力の和(つまり3本の入力に出てくる1の数)を求めて、2進表示し、1の桁(右側の桁)をZとし、2の桁(左側の桁)を上への繰上りとすればよい。 入力欄の方は、3本の入力の0／1のすべてのパターン(000から111まで)を収容すること。のこの表は、暗記するのではなく、今述べた原理に従って自分で作成できること。

入力			出力	
X	Y	繰上り	Z	繰上り
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

d.　下の図



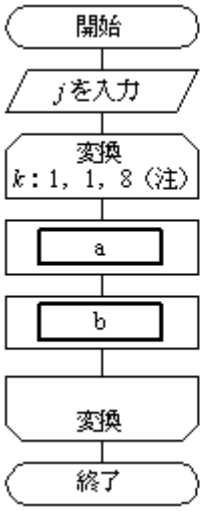
e.　繰上りが下の桁から順次伝播する回路であり、下の桁の繰上り状況が決まらないと次の桁の出力と繰上りが決まらない。右端の桁（1の桁）から順に左へ繰上りが伝わっていくので、この繰上り信号の伝播の遅延が問題になる。 たとえば1段（1桁）当たり伝搬に 10 ピコ秒（1 ピコ秒＝10^{-12}秒）かかるとすると、32ビットの加算回路では 310 ピコ秒かかる。
f.　「キャリールックアヘッド回路」と呼ばれる、全ての桁の入力からそれぞれの桁の繰上り信号の値を（1段の遅延時間で）計算する回路が使われている。

［2］
a.　B_0～B_3 の入力を、一方は S_0 と AND を取って OR へ、他方は NOT を通ってから S_1 と AND を取って OR へ行っている。この部分である。
b.　1を加えたいときに、最下位ビット（ビット0）へ繰上り入力信号 C_0＝1を加えることによって、実現している。通常の足し算の場合には、「最下位ビットへの繰上り入力信号」 C_0 は0にしておく。 しかし2の補数を計算するときは C_0＝1を加える。

［3］
a.　① 792　1848　264　45672
② 2段目の第2桁の7は、掛ける数の中で $70 = 7 \times 10^1$ を表す。 従って、$264 \times (7 \times 10^1) = (264 \times 7) \times 10^1$ であり、264×7を10倍するために左へ一桁ずらしているのである。 同様に、3段目の第3桁1の 1 は、$100 = 1 \times 10^2$ を表す。従って、$264 \times (1 \times 10^2) = (264 \times 1) \times 100$。
b.　① 0010　0000　0010　0000　001010
② 上記1）の②と同じだが、10^1、10^2 の代わりに 2^2 を入れることになる。
c.　②が正しい。
d.　略
e.　① 00001　② 00001　③ 00101　④ 11101
f.　① $01001000_2 = 72_{10}$　41を8倍するので328だが、8ビットなので桁あふれを起こし256が失われて328-256＝72。
② $00000101_2 = 5_{10}$　41を8で割るので5になる。
③ ②と同じ
g.　① $10110000_2 = 176_{10}$　$11010110_2 = -42_{10}$ を8倍すると -336_{10} だが、シフト結果とは一致しない（正しくない）。符号なしで考えれば $10110000_2 = 176_{10}$ 、 $11010110_2 = 214_{10}$ を 8 倍すると 1712_{10} だが、桁あふれの結果 1536 が失われて、1712-1536=176。
② $00011010_2 = 26_{10}$　符号が正しくない上、値も-42を8で割った結果の-5とは一致しない（正しくない）。
③ $11111010_2 = -6_{10}$　-42を8で割った結果は-5.1であるが、整数演算なので切捨てられて-5.1は-6になるので、正しい。
［文字コード］
① 5ビット　2^5＝32　② 4ビット　2^4＝16　③ 5＋4＝9ビット　④ 6ビット　2^6＝64
⑤ 合計 10+26+26+33＝95 文字　それを表すには7ビット必要
⑥ 95+33＝128　それを表すには7ビットすべてを使う

≪基本情報処理技術者試験問題から関連問題≫

1） 次の流れ図は，10 進整数 j（0<j<100)を2進数に変換する処理を表している。2進数は下位けたから順に，配列の要素 NISHIN(1)から NISHIN(8) に格納される。流れ図の a 及び b に入る処理はどれか。ここで，j div 2 は j を2で割った商の整数部分を，j mod 2 は j を2で割った余りを表す。（基本 17 春 01）



（注）ループ端の繰返し指定は，
変数名：初期値，増分，終値
を示す。

	a	b
ア	j div 2 → j	j mod 2 → NISHIN(k)
イ	j div 2 → NISHIN(k)	j mod 2 → j
ウ	j mod 2 → j	j div 2 → NISHIN(k)
エ	j mod 2 → NISHIN(k)	j div 2 → j

2） 10 進数の分数 1/32 を 16 進数の小数で表したものはどれか。（基本 20 春 02）

ア 0.01　　イ 0.02　　ウ 0.05　　エ 0.08

3） 16 進小数 0.FEDC を4倍したものはどれか。（基本 16 秋 03）

ア 1.FDB8　　イ 2.FB78　　ウ 3.FB70　　エ F.EDC0

4） 16 進小数 2A.4C と等しいものはどれか。（基本 18 春 01）

ア $2^5+2^3+2^1+2^{-2}+2^{-5}+2^{-6}$　　イ $2^5+2^3+2^1+2^{-1}+2^{-4}+2^{-5}$

ウ $2^6+2^4+2^2+2^{-2}+2^{-5}+2^{-6}$　　エ $2^6+2^4+2^2+2^{-1}+2^{-4}+2^{-5}$

5） 16 進小数 3A . 5C を 10 進数の分数で表したものはどれか。（基本 19 春 01）

ア 939/16　　イ 3735/64　　ウ 14939/256　　エ 14941/256

6） 16 進数の小数 0.248 を 10 進数の分数で表したものはどれか。（基本 16 秋 01）

ア 31/32　　イ 31/125　　ウ 31/512　　エ 73/512

7） 16 進小数 0.C を 10 進小数に変換したものはどれか。(基本 19 秋 01)

ア 0.12　　イ 0.55　　ウ 0.75　　エ 0.84

8） 次の 10 進小数のうち，8進数に変換したときに有限小数になるものはどれか。（基本 17 秋 01）

ア 0.3　　イ 0.4　　ウ 0.5　　エ 0.8

9） 次の計算は何進法で成立するか。（基本 14 秋 02)(基本 18 春 02)

131 － 45 ＝ 53

ア 6　　イ 7　　ウ 8　　エ 9

10） 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

11） 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。

12） コンピュータを使用して整数の加減算を行う場合，あふれ（オーバフロー）に留意する必要がある。あふれの可能性がある演算をすべて列記したものはどれか。（基本 15 秋 04）

	演算	オペランド x	オペランド y
a	x ＋ y	正	正
b	x ＋ y	正	負
c	x ＋ y	負	正
d	x ＋ y	負	負
e	x － y	正	正
f	x － y	正	負
g	x － y	負	正
h	x － y	負	負

ア a，d，f，g　　イ b，c，e，h　　ウ b，e　　エ c，e，h

13） 数値を2進数で格納するレジスタがある。このレジスタに正の整数 x を入れた後，“レジスタの値を2ビット左にシフトして，これに x を加える”操作を行うと，レジスタの値は x の何倍になるか。ここで，シフトによるあふれ（オーバフロー）は，発生しないものとする。（基本 15 春 02）（基本 18 秋 02）

ア 3　　イ 4　　ウ 5　　エ 6

14） 数多くの数値の加算を行う場合，絶対値の小さなものから順番に計算するとよい。これはどの誤差を抑制する方法を述べたものか。（基本 14 秋 04）（基本 17 春 04）

ア アンダフロー　　イ 打切り誤差　　ウ けた落ち　　エ 情報落ち

15） 浮動小数点演算において、絶対値の大きな数と絶対値の小さな数の加減算を行ったとき、絶対値の小さな数の有効けたの一部又は全部が結果に反映されないことを何というか。（基本 15 春 04）（基本 20 秋 04）

ア 打切り誤差　　イ けた落ち　　ウ 情報落ち　　エ 絶対誤差

16） 浮動小数点表示の仮数部が 23 ビットであるコンピュータで計算した場合，情報落ちが発生する計算式はどれか。ここで，( )2 内の数は2進法で表示されている。（基本 18 春 05）（基本 20 春 05）

ア $(10.101)_2 \times 2^{-16} - (1.001)_2 \times 2^{-15}$　　イ $(10.101)_2 \times 2^{16} - (1.001)_2 \times 2^{16}$

ウ $(1.01)_2 \times 2^{18} + (1.01)_2 \times 2^{-5}$　　エ $(1.001)_2 \times 2^{20} + (1.1111)_2 \times 2^{21}$

17） 32 ビットのレジスタに 16 進数 ABCD が入っているとき，2ビットだけ右に論理シフトしたときの値はどれか。（基本 16 春 04）

ア 2AF3　　イ 6AF3　　ウ AF34　　エ EAF3

18） 数値を2進数で表すレジスタがある。このレジスタに格納されている正の整数xを 10 倍する操作はどれか。ここで，シフトによるけたあふれは，起こらないものとする。（基本 20 春 04）

ア xを2ビット左にシフトした値にxを加算し，更に1ビット左にシフトする。

イ xを2ビット左にシフトした値にxを加算し，更に2ビット左にシフトする。

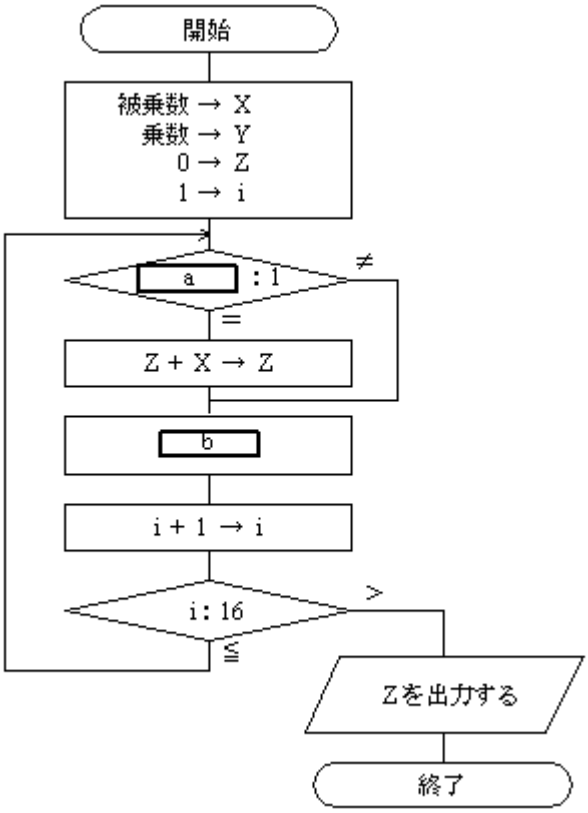
ウ xを3ビット左にシフトした値と，xを2ビット左にシフトした値を加算する。

エ xを3ビット左にシフトした値にxを加算し，更に1ビット左にシフトする。

19） 整数 m がレジスタに2進数として入っている。これを3ビット左にシフトしたものに m を加えると，結果は元の m の何倍になるか。ここで，あふれが生じることはないものとする。（基本 17 秋 03）

ア 4　　イ 7　　ウ 8　　エ 9

20） 次の流れ図は，シフト演算と加算の繰返しによって2進数の乗算を行う手順を表したものである。この流れ図中の a，b の処理の組合せとして，正しいものはどれか。ここで，乗数と被乗数は符号なしの 16 ビットで表される。X，Y，Z は 32 ビットのレジスタであり，けた送りには論理シフトを用いる。（基本 18 春 15）



	a	b
ア	Y の最下位ビット	X を1ビット左シフト，Y を1ビット右シフト
イ	Y の最下位ビット	X を1ビット右シフト，Y を1ビット左シフト
ウ	Y の最上位ビット	X を1ビット左シフト，Y を1ビット右シフト
エ	Y の最上位ビット	X を1ビット右シフト，Y を1ビット左シフト

21） 浮動小数点形式で表現される数値の演算において，有効けた数が大きく減少するものはどれか。（基本 14 春 04）

ア 絶対値がほぼ等しく，同符号である数値の加算

イ 絶対値がほぼ等しく，同符号である数値の減算

ウ 絶対値の大きな数と絶対値の小さな数との絶対値による加算

エ 絶対値の大きな数と絶対値の小さな数との絶対値による減算

22） けた落ちの説明として，適切なものはどれか。（基本 16 春 05）

ア 値がほぼ等しい浮動小数点数同士の減算において，有効けた数が大幅に減ってしまうことである。

イ 演算結果が，扱える数値の最大値を超えることによって生じる誤差である。

ウ 数表現のけた数に限度があるとき，最小のけたより小さい部分について四捨五入，切上げ又は切捨てを行うことによって生じる誤差である。

エ 浮動小数点数の加算において，一方の数値の下位のけたが欠落することである。

23） 8ビットのレジスタがある。このレジスタの各ビットの値を d 0、d 1, …, d 7 とし、パリティビットの値を p とする。奇数パリティの場合、常に成立する関係式はどれか。ここで、⊕は排他的論理和演算を表す。（基本 15 春 09）（基本 19 春 10）

ア $0 \oplus d_0 \oplus d_1 \oplus \cdots \oplus d_7 = p$

イ $d_0 \oplus d_1 \oplus \cdots \oplus d_7 = p$

ウ $d_0 \oplus d_1 \oplus \cdots \oplus d_7 \oplus p = 0$

エ $d_0 \oplus d_1 \oplus \cdots \oplus d_7 \oplus p = 1$

≪以下は文字コードの問題である。挑戦してみたらよいだろう≫

24） 文字列“ET”を ASCII でコード化したものを 16 進表記したものはどれか。ここで，文字コードの8ビット目には，偶数パリティビットが付く。（基本 19 春 11）

〔ASCII コード表の一部〕

b7	b6	b5	b4	b3	b2	b1		1	1
								0	0
								0	1
			0	0	0	0		@	P
			0	0	0	1		A	Q
			0	0	1	0		B	R
			0	0	1	1		C	S
			0	1	0	0		D	T
			0	1	0	1		E	U
			0	1	1	0		F	V
			0	1	1	1		G	W

ア 4554　　イ A32B　　ウ ACA5　　エ C5D4

25） 次の表は JIS コード表の一部である。二つの文字“A”と“2”をこの順に JIS コードで表したものはどれか。（基本 18 春 08）

b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	列 / 行	0	0	0	0
									0	0	0	1
									0	1	1	0
									1	0	1	0
									1	2	3	4
				0	0	0	1	1			1	A
				0	0	1	0	2			2	B
				0	0	1	1	3			3	C

ア 00010100 00100011　　イ 00110010 01000001

ウ 01000001 00110010　　エ 01000010 00110010

26） コンピュータで使われている文字符号の説明のうち，適切なものはどれか。（基本 16 秋 73）（基本 18 春 69）

ア ASCII 符号はアルファベット，数字，特殊文字及び制御文字からなり，漢字に関する規定はない。

イ EUC は文字符号の世界標準を作成しようとして考案された 16 ビット以上の符号体系であり，漢字に関する規定はない。

ウ Unicode は文字の1バイト目で漢字かどうかが分かるようにする目的で制定され，漢字を ASCII 符号と混在可能とした符号体系である。

エ シフト JIS 符号は UNIX における多言語対応の一環として制定され，ISO として標準化されている。

27） Unicode の説明として，適切なものはどれか。（基本 16 春 73）

ア ANSI（米国標準規格協会）で定めた，7ビットの文字コード体系である。

イ JIS（日本工業規格）で定めた文字コード体系であり，英数字とカタカナを扱う8ビットのコードと，全角文字を扱う16 ビットのコードがある。

ウ 拡張 UNIX コードとも呼ばれ，全角文字と半角カタカナ文字を2バイト又は3バイトで表現する。

エ 多国籍文字を扱うために，日本語や中国語などの形の似た文字を同一コードに割り当てて 2 バイトの文字コードで表現する。

28） UCS-2（Unicode）を説明したものはどれか。（基本 19 春 69）

ア JIS から派生したコード体系であり，英数字は1バイト，漢字は2バイトで表現する。

イ 主に UNIX で使用するコード体系であり，英数字は1バイト，漢字は2バイトで表現する。

ウ すべての文字を1バイトで表現するコード体系である。

エ すべての文字を2バイトで表現するコード体系であり，多くの国の文字体系に対応できる。

29） 英字の大文字（ A ～ Z ）と数字（ 0 ～ 9 ）を同一のビット数で一意にコード化するには，少なくとも何ビット必要か。（基本 15 秋 09）

ア 5　　イ 6　　ウ 7　　エ 8

30） 7ビットの文字コードの先頭に1ビットの偶数パリティビットを付加するとき，文字コード 30，3F，7A にパリティビットを付加したものはどれか。ここで，文字コードは 16 進数で表している。（基本 14 秋 09）（基本 17 春 10）（基本 20 春 10）

ア 30，3F，7A　　イ 30，3F，FA　　ウ B0，3F，FA　　エ B0，BF，7A

≪解答≫

1） エ　2） エ　3） ウ　4） ア　5） イ　6） エ　7） ウ　8） ウ　9） イ　10） イ

11） ア　12） ア　13） ウ　14） エ　15） ウ　16） ウ　17） ア　18） ア　19） エ　20） ア

21） イ　22） ア　23） エ　24） エ　25） ウ　26） ア　27） エ　28） エ　29） イ　30） イ