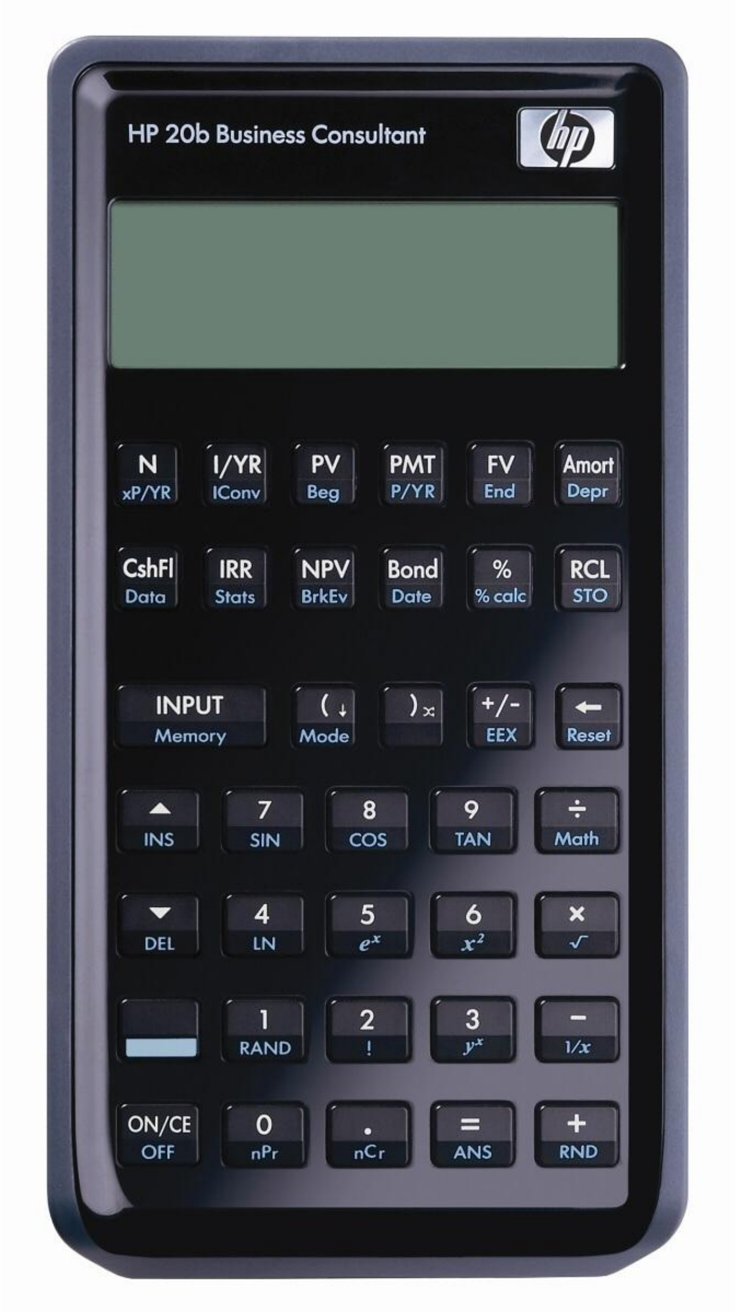
CALCULATOR HP 20b

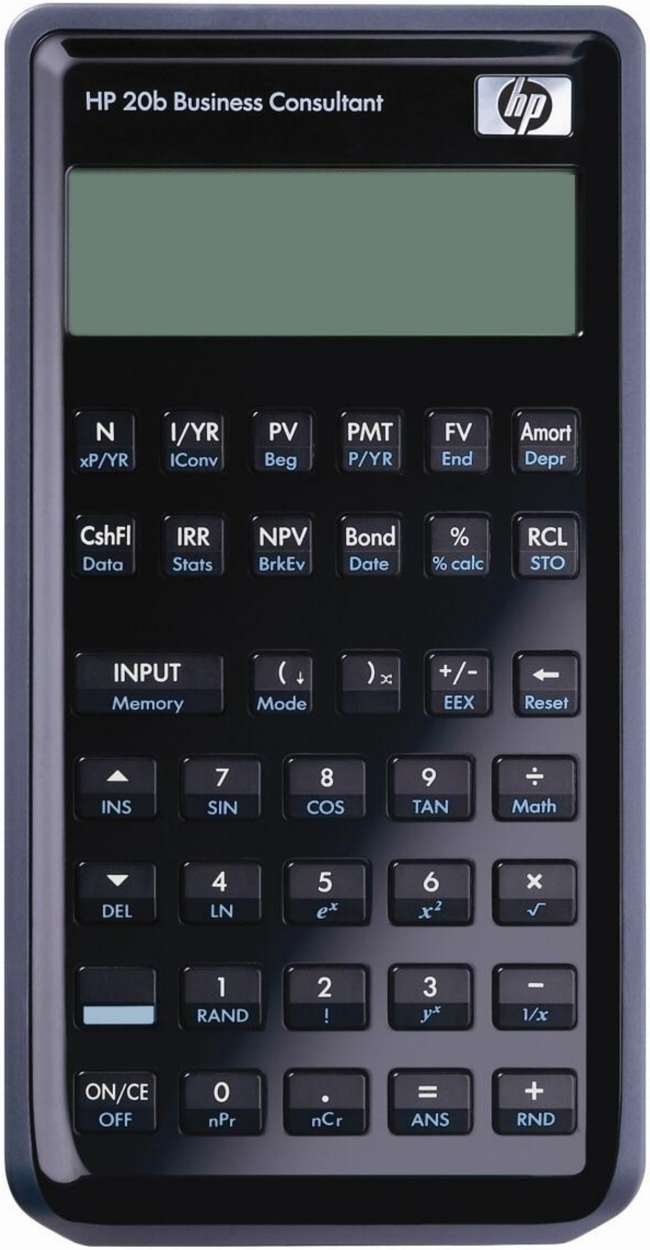
Aaron Burger and Isabel Baransky

From RPN to beyond

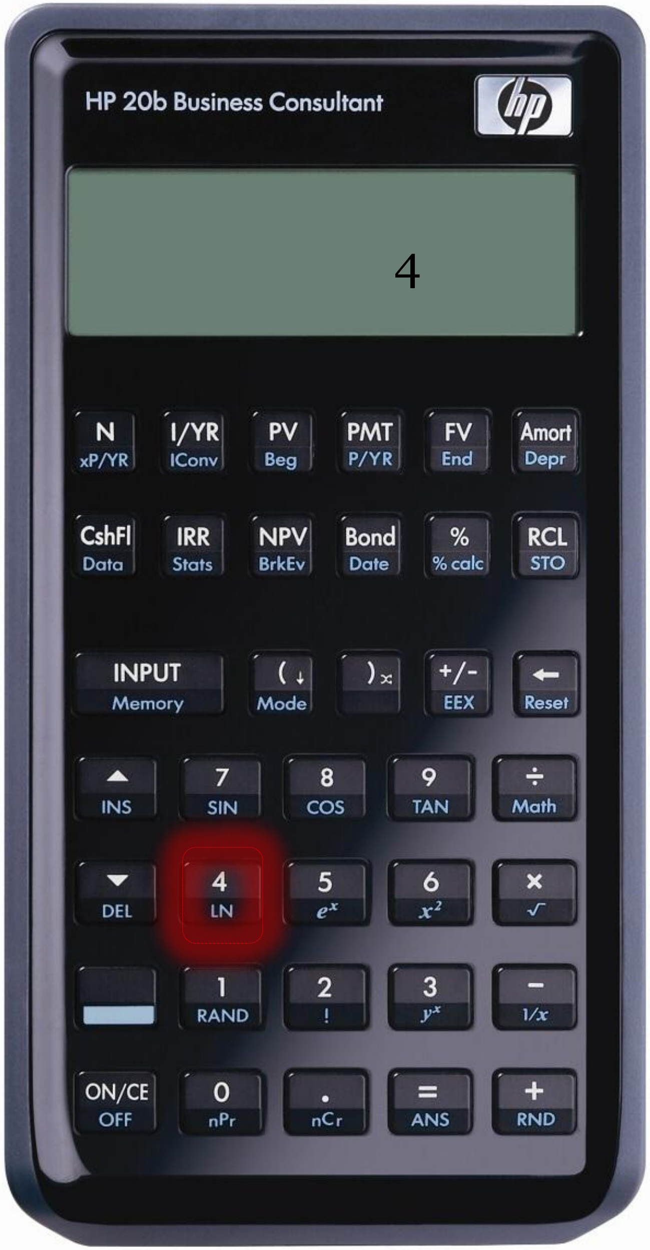


HP 20b CALCULATOR

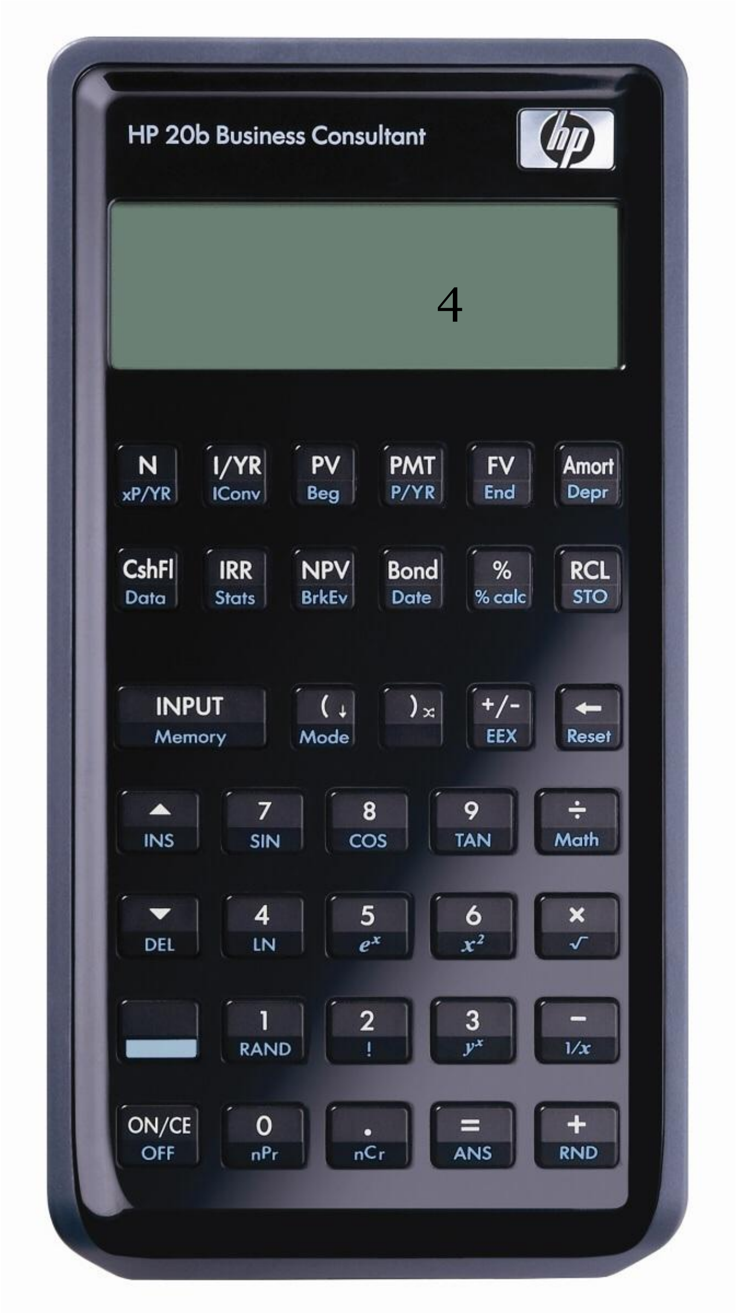
HOW TO USE


HP 20b Business Consultant
hp
N xP/YR
I/YR IConv
PV Beg
PMT P/YR
FV End
Amort Depr
CshFl Data
IRR Stats
NPV BrkEv
Bond Date
% % calc
RCL STO
INPUT Memory
(Mode
)
+/- EEX
← Reset
INS
7 SIN
8 COS
9 TAN
÷ Math
DEL
4 LN
5 e^x
6 x²
× √
1 RAND
2 !
3 y^x
− 1/x
ON/CE OFF
0 nPr
. nCr
= ANS
+ RND

DISPLAY

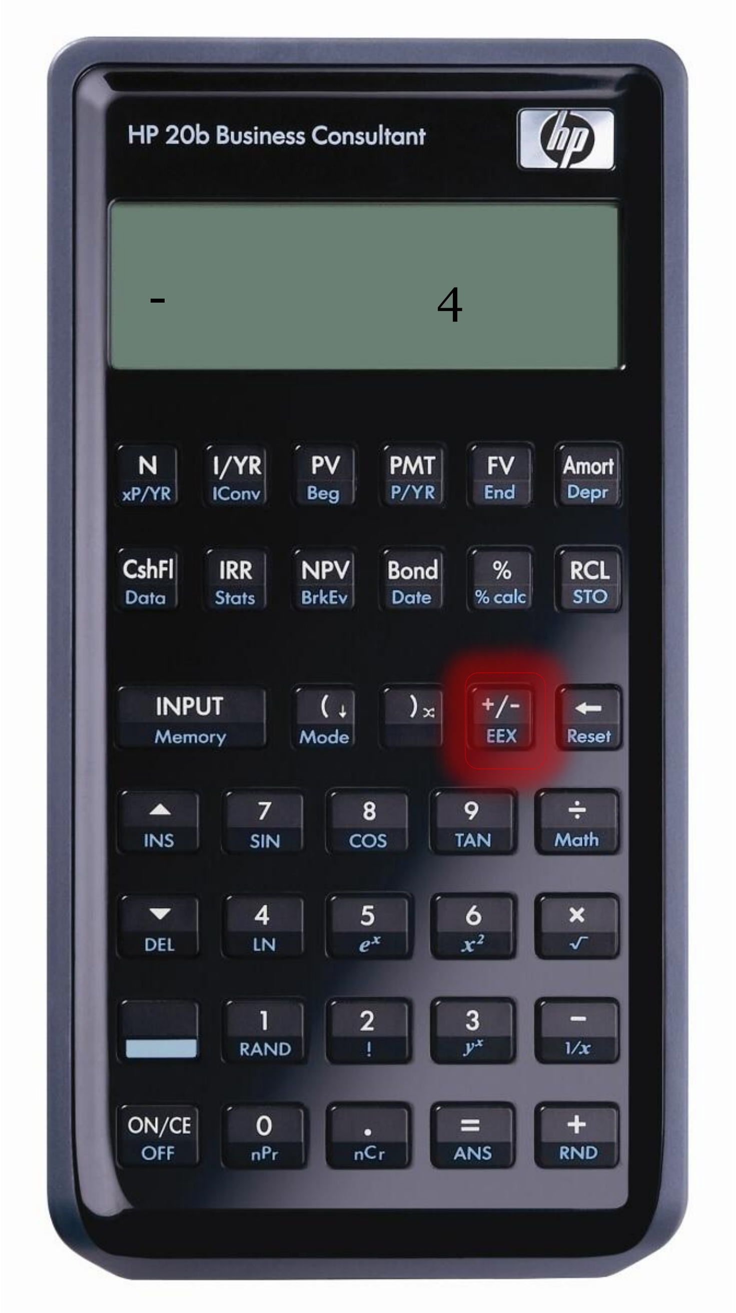

HP 20b Business Consultant
hp
4
N xP/YR
I/YR IConv
PV Beg
PMT P/YR
FV End
Amort Depr
CshFl Data
IRR Stats
NPV BrkEv
Bond Date
% % calc
RCL STO
INPUT Memory
(Mode
)
+/- EEX
← Reset
INS
7 SIN
8 COS
9 TAN
÷ Math
DEL
4 LN
5 e^x
6 x²
× √
1 RAND
2 !
3 y^x
− 1/x
ON/CE OFF
0 nPr
. nCr
= ANS
+ RND

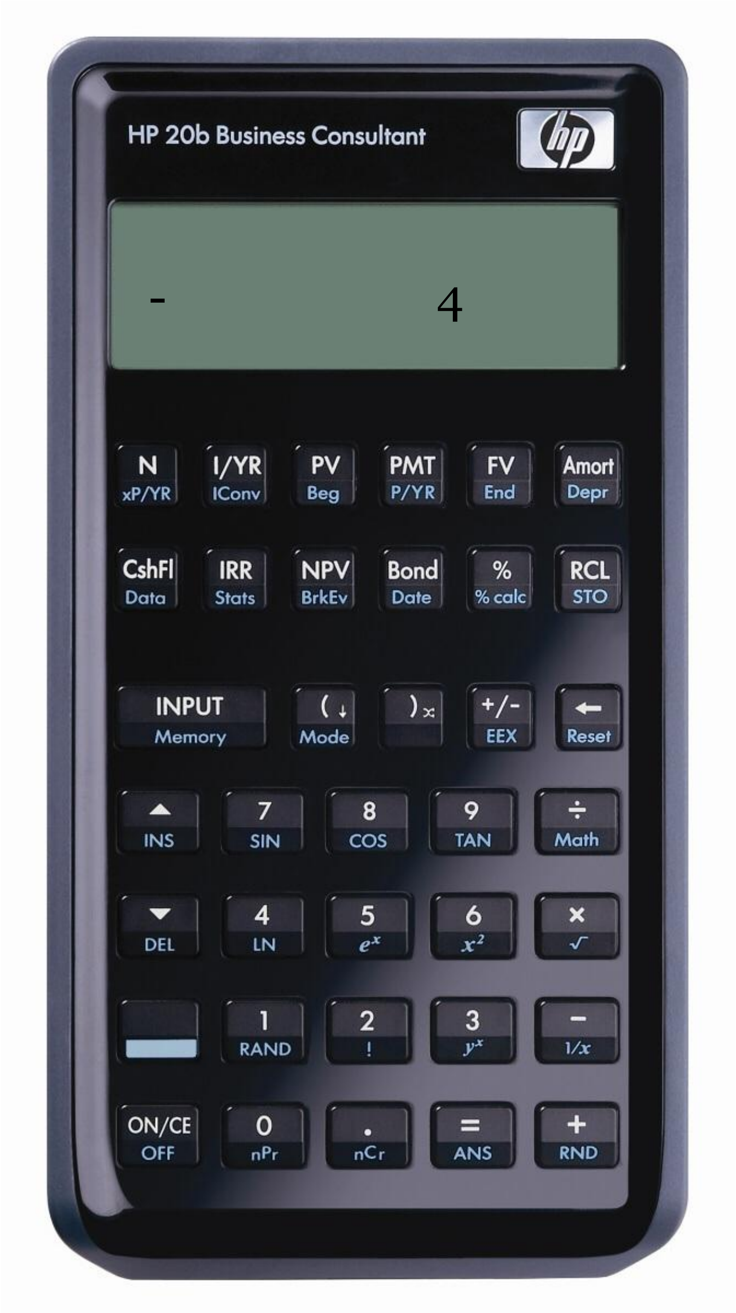
DISPLAY


HP 20b Business Consultant
hp
4
N xP/YR
I/YR IConv
PV Beg
PMT P/YR
FV End
Amort Depr
CshFl Data
IRR Stats
NPV BrkEv
Bond Date
% % calc
RCL STO
INPUT Memory
(Mode
)
+/- EEX
← Reset
INS
7 SIN
8 COS
9 TAN
÷ Math
DEL
4 LN
5 e^x
6 x²
× √
1 RAND
2 !
3 y^x
− 1/x
ON/CE OFF
0 nPr
. nCr
= ANS
+ RND

DISPLAY

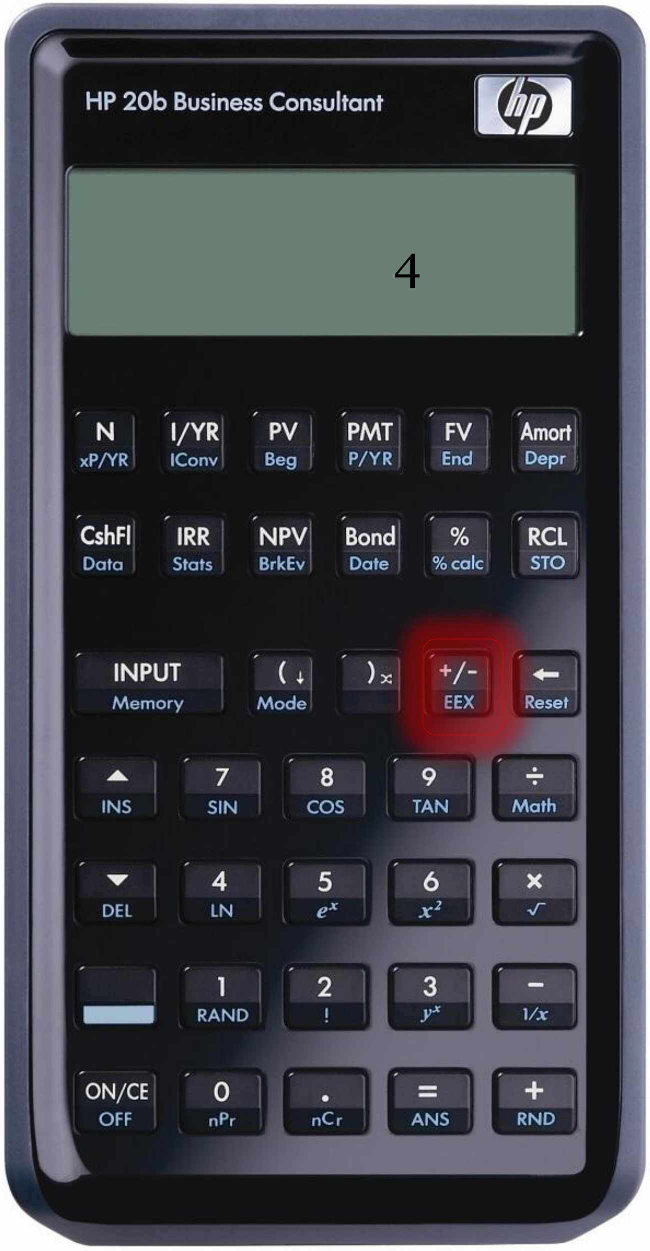
SIGN CHANGE

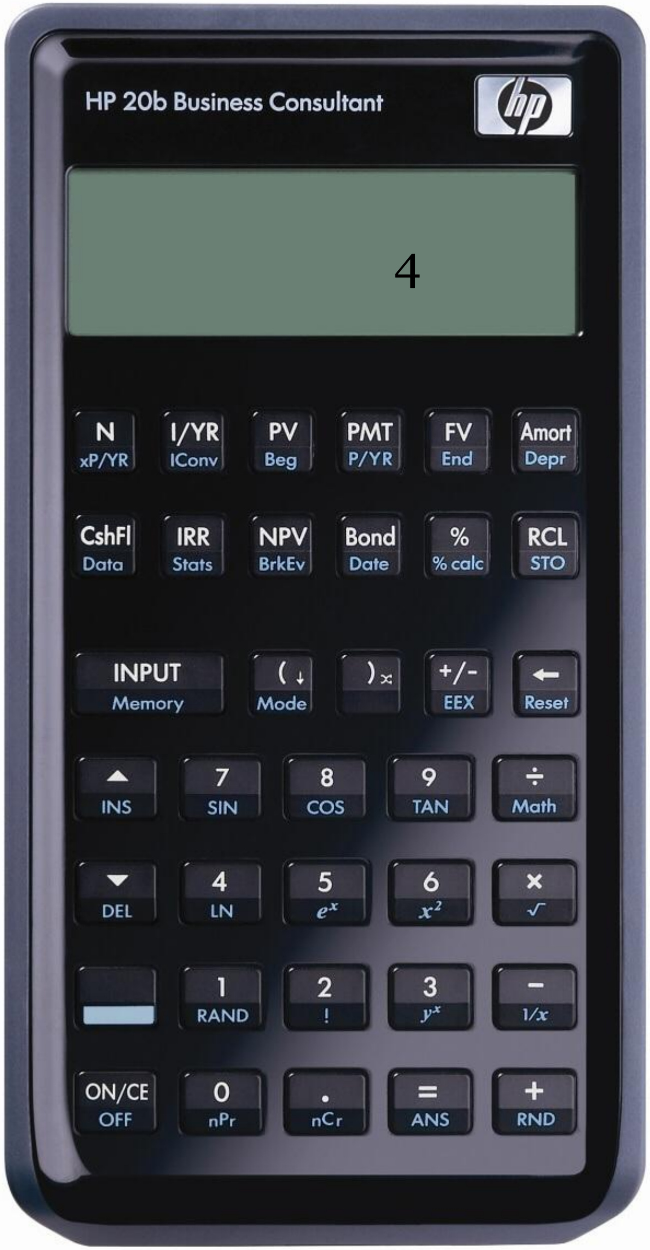



HP 20b Business Consultant
hp
-
4
N
xP/YR
I/YR
IConv
PV
Beg
PMT
P/YR
FV
End
Amort
Depr
CshFl
Data
IRR
Stats
NPV
BrkEv
Bond
Date
%
% calc
RCL
STO
INPUT
Memory
(↓
Mode
) x
+/-
EEX
←
Reset
▲
INS
7
SIN
8
COS
9
TAN
÷
Math
▼
DEL
4
LN
5
e^x
6
x²
×
√
1
RAND
2
!
3
y^x
−
1/x
ON/CE
OFF
0
nPr
.
nCr
=
ANS
+
RND

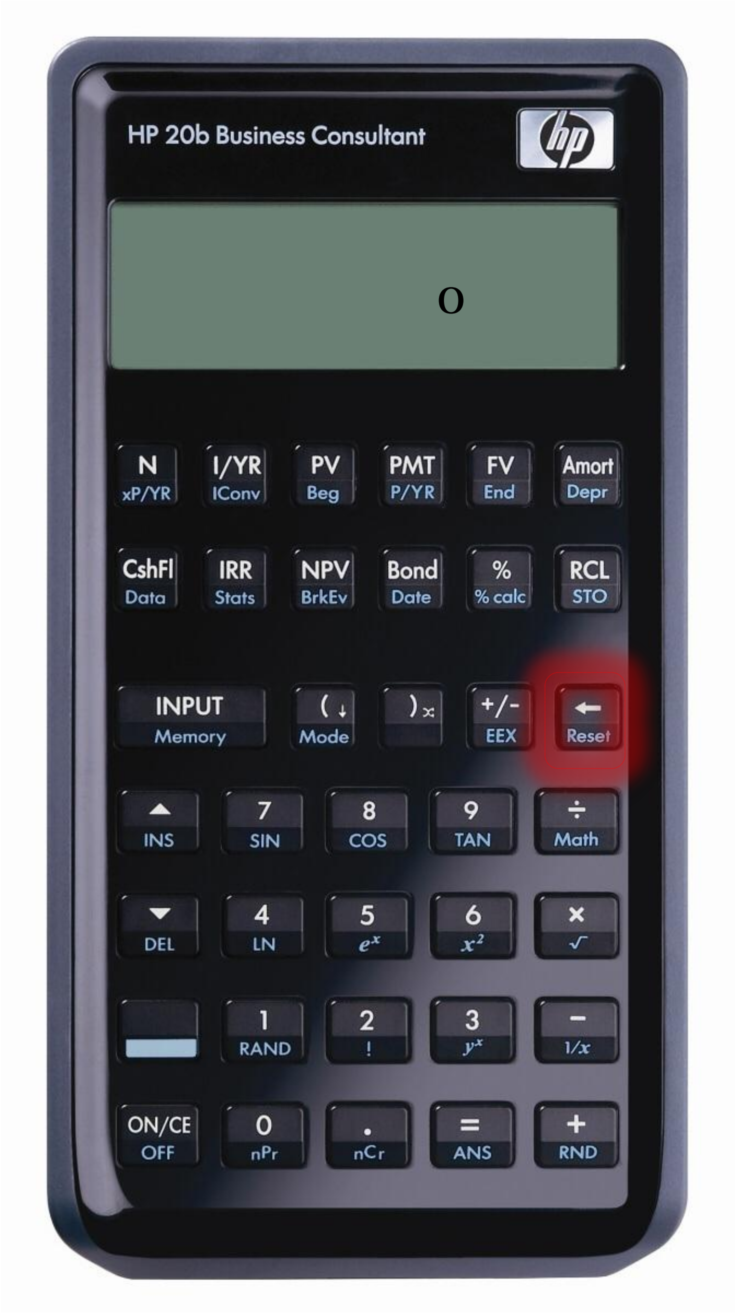
SIGN CHANGE

SIGN CHANGE


HP 20b Business Consultant
hp
4
N xP/YR
I/YR IConv
PV Beg
PMT P/YR
FV End
Amort Depr
CshFl Data
IRR Stats
NPV BrkEv
Bond Date
% % calc
RCL STO
INPUT Memory
(↓ Mode
) x
+/- EEX
← Reset
▲ INS
7 SIN
8 COS
9 TAN
÷ Math
▼ DEL
4 LN
5 e^x
6 x^2
× √
1 RAND
2 !
3 y^x
− 1/x
ON/CE OFF
0 nPr
. nCr
= ANS
+ RND

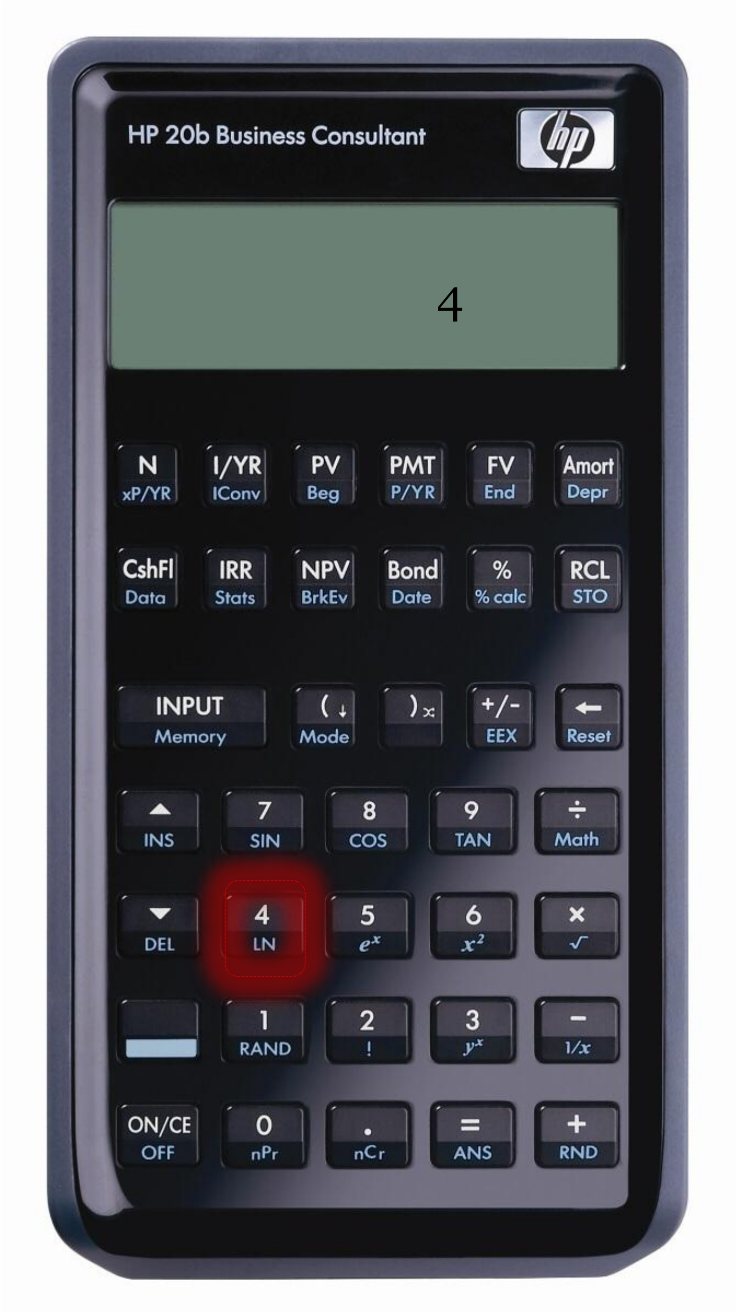

HP 20b Business Consultant
hp
4
N xP/YR
I/YR IConv
PV Beg
PMT P/YR
FV End
Amort Depr
CshFl Data
IRR Stats
NPV BrkEv
Bond Date
% % calc
RCL STO
INPUT Memory
(↓ Mode
) x
+/- EEX
← Reset
▲ INS
7 SIN
8 COS
9 TAN
÷ Math
▼ DEL
4 LN
5 e^x
6 x²
× √
1 RAND
2 !
3 y^x
− 1/x
ON/CE OFF
0 nPr
. nCr
= ANS
+ RND

SIGN CHANGE


HP 20b Business Consultant
hp
0
N xP/YR
I/YR IConv
PV Beg
PMT P/YR
FV End
Amort Depr
CshFl Data
IRR Stats
NPV BrkEv
Bond Date
% % calc
RCL STO
INPUT Memory
(↓ Mode
) x
+/- EEX
← Reset
INS
7 SIN
8 COS
9 TAN
÷ Math
DEL
4 LN
5 e^x
6 x^2
× √
1 RAND
2 !
3 y^x
− 1/x
ON/CE OFF
0 nPr
. nCr
= ANS
+ RND

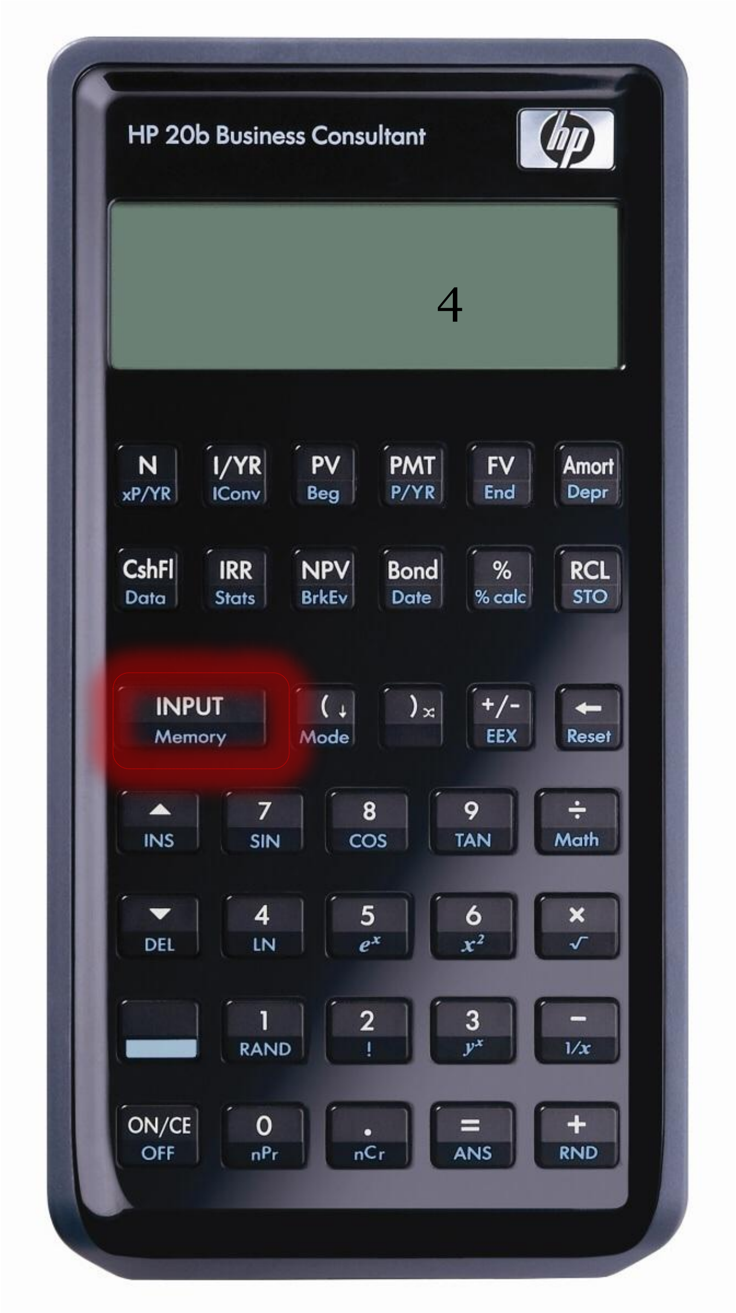
RESET

ALGEBRA



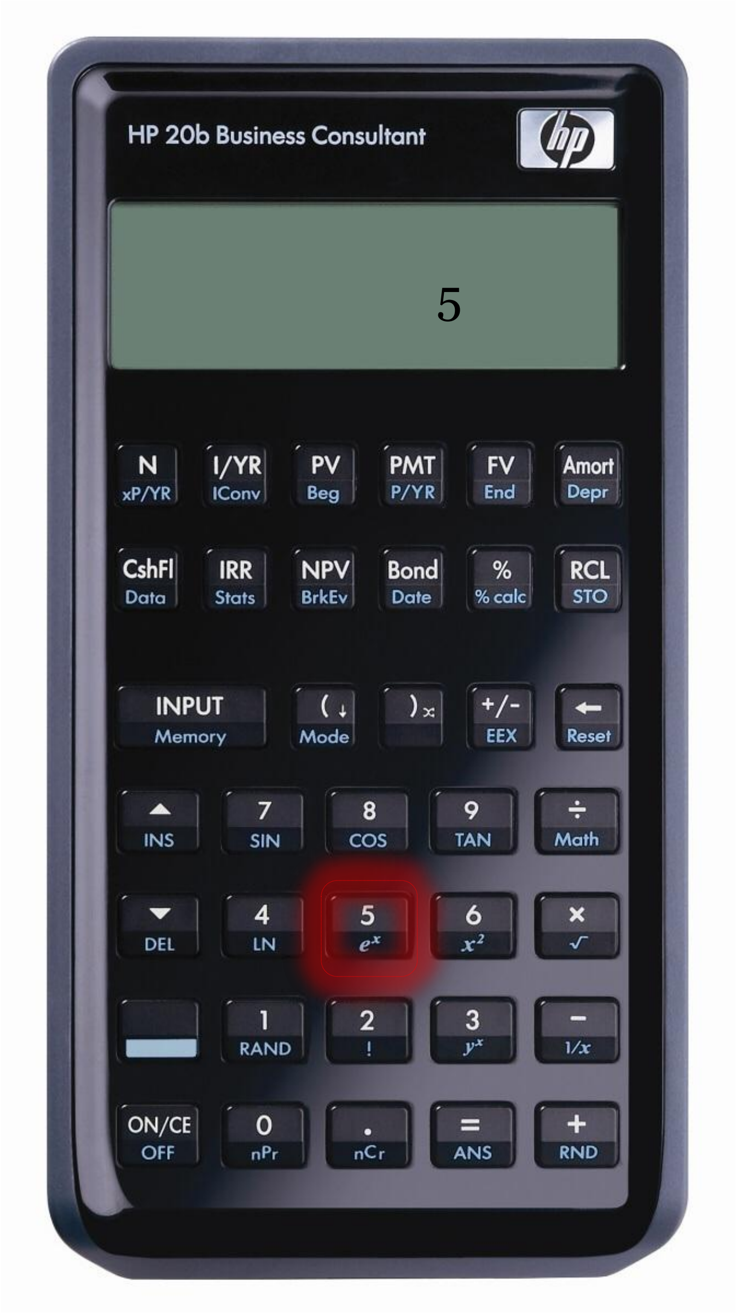
(4+5) X 2

ALGEBRA



(4+5) X 2

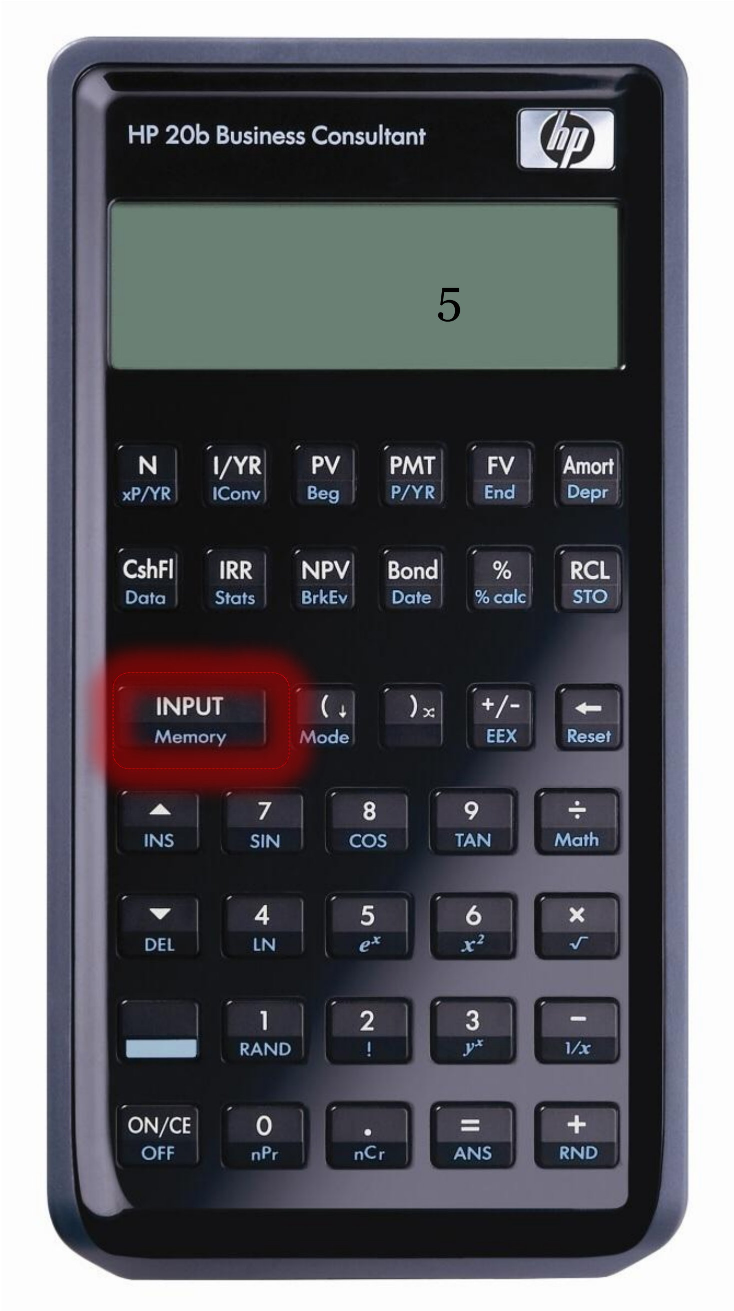
4



ALGEBRA

(4+5) X 2

4

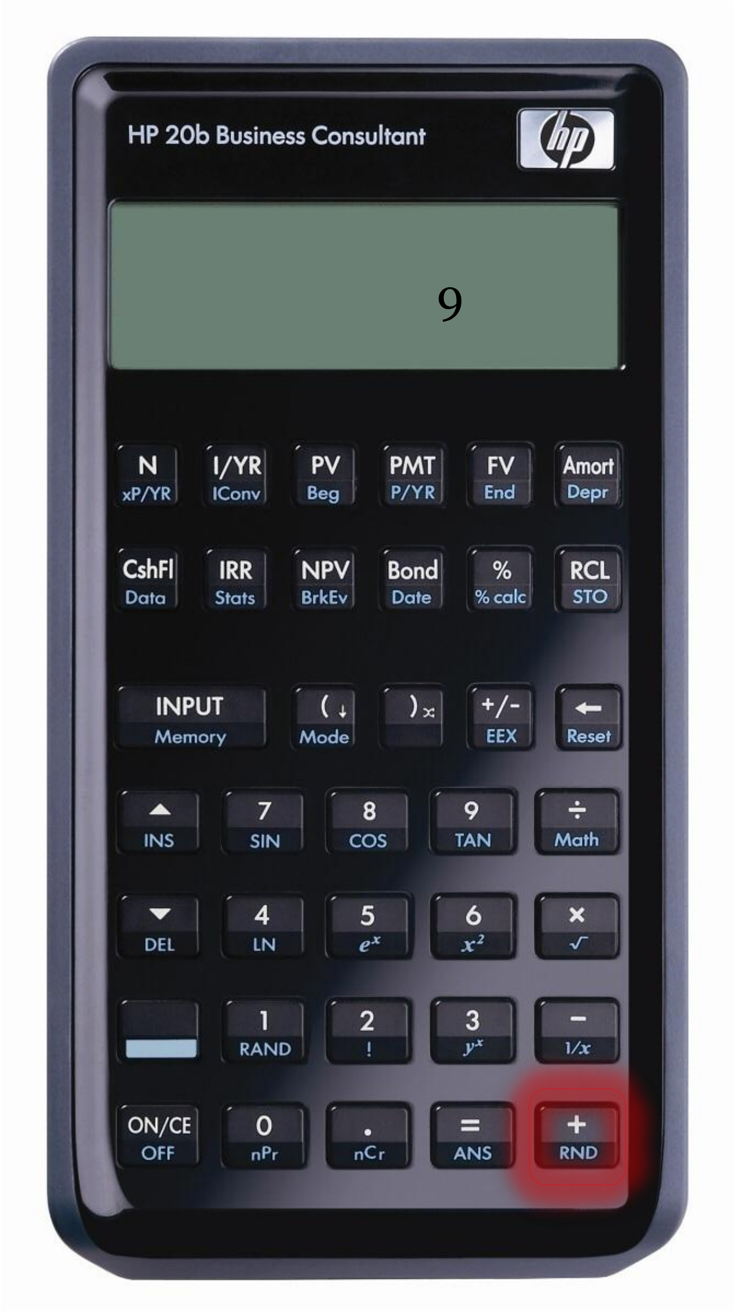


ALGEBRA

(4+5) X 2

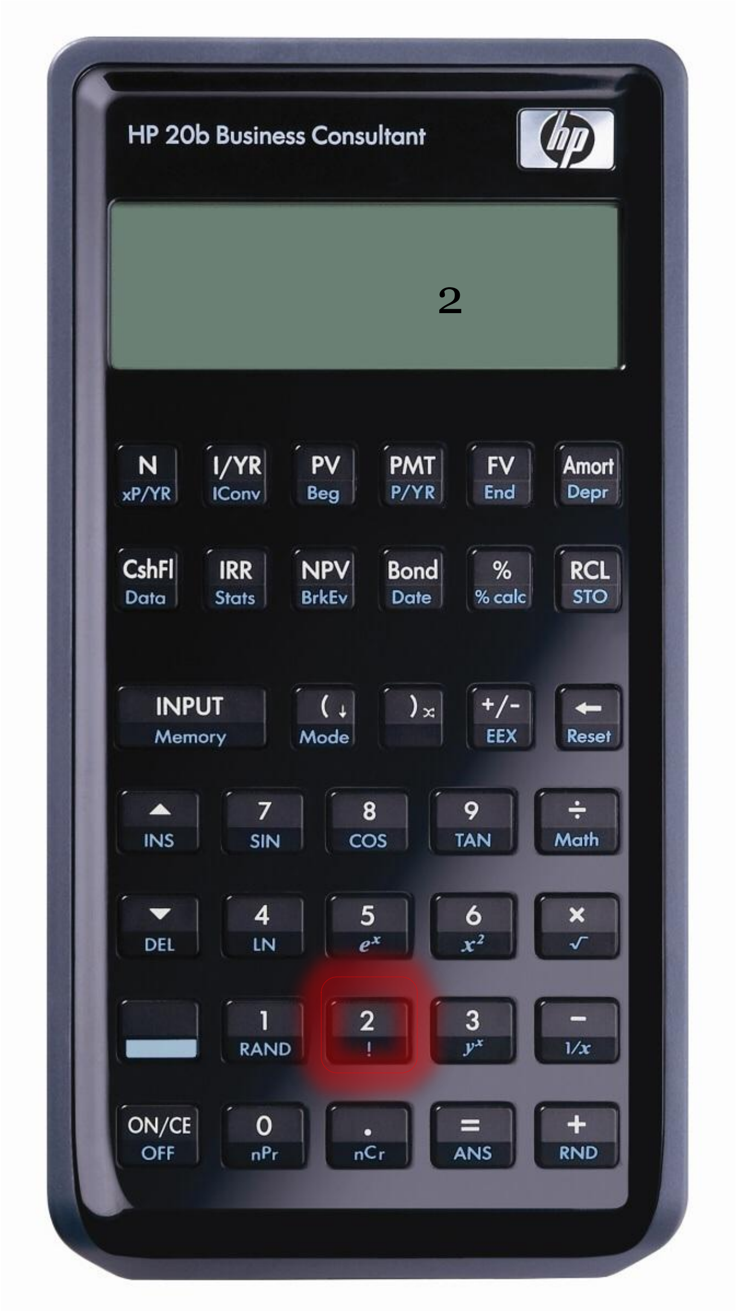
4; 5

ALGEBRA



(4+5) X 2

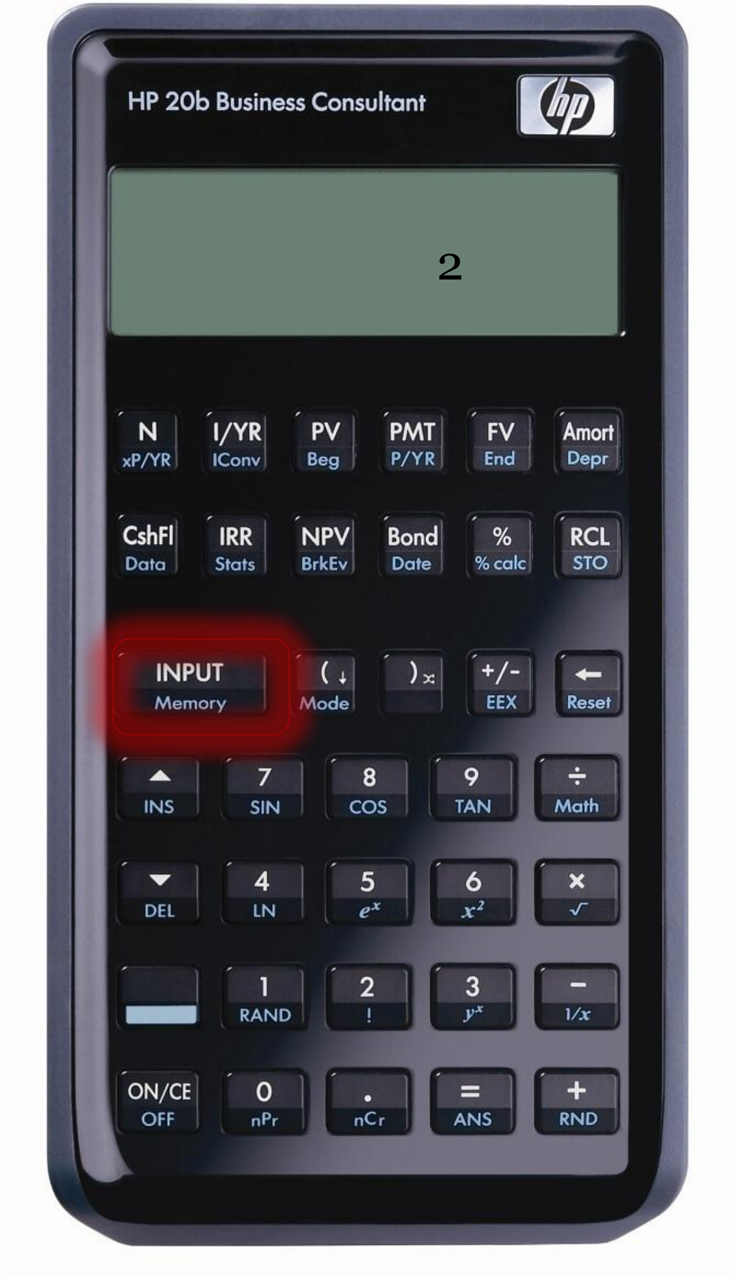
9



ALGEBRA

(4+5) X 2

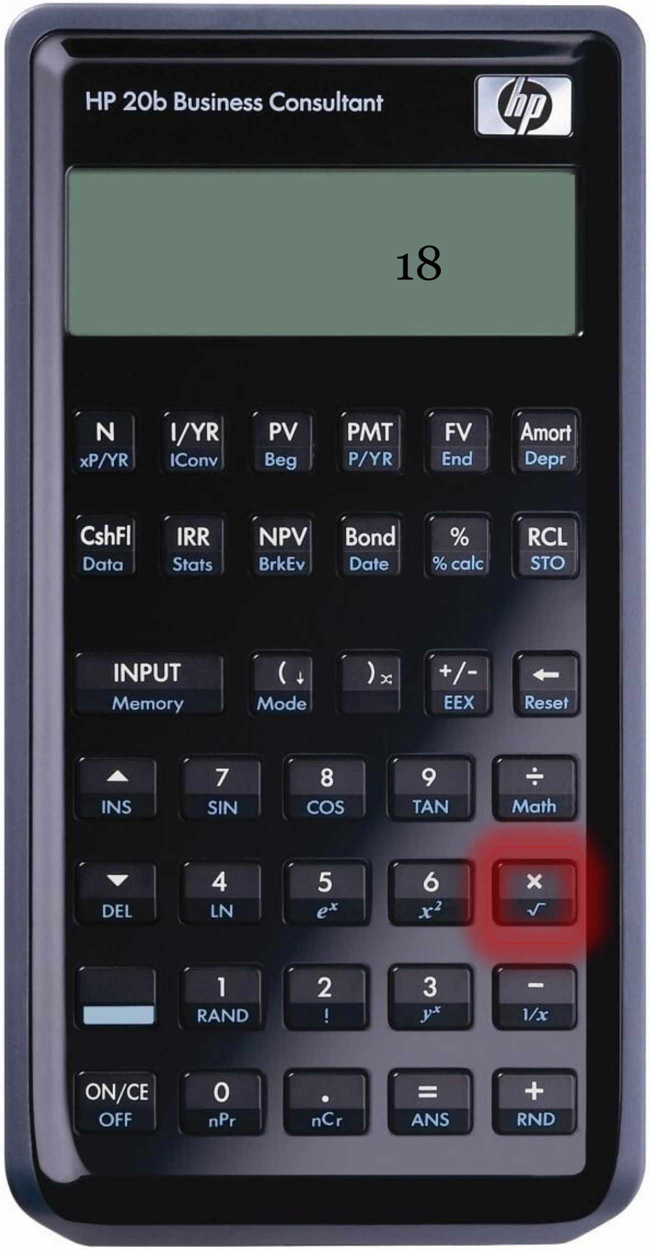
9



ALGEBRA

(4+5) X 2

9; 2



ALGEBRA

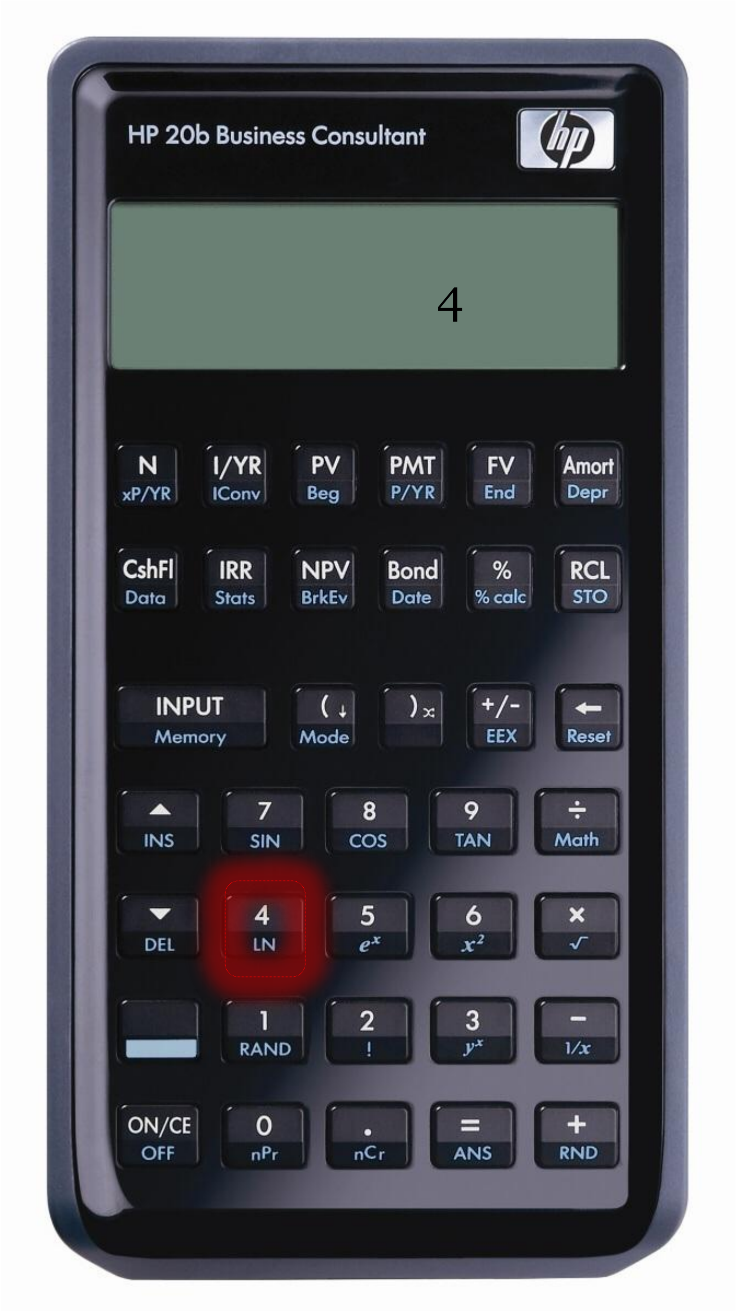
(4+5) X 2

18

WITH IMPROVEMENTS

ALGEBRA

4+5 X 2

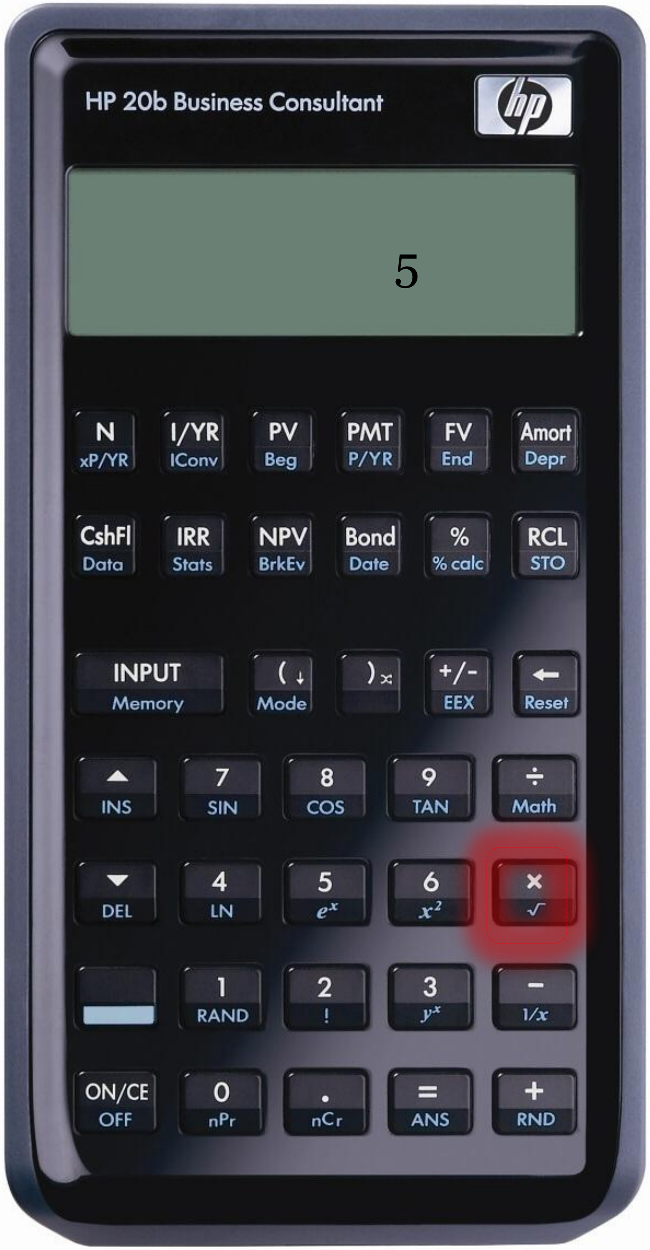


ALGEBRA

4+5 X 2

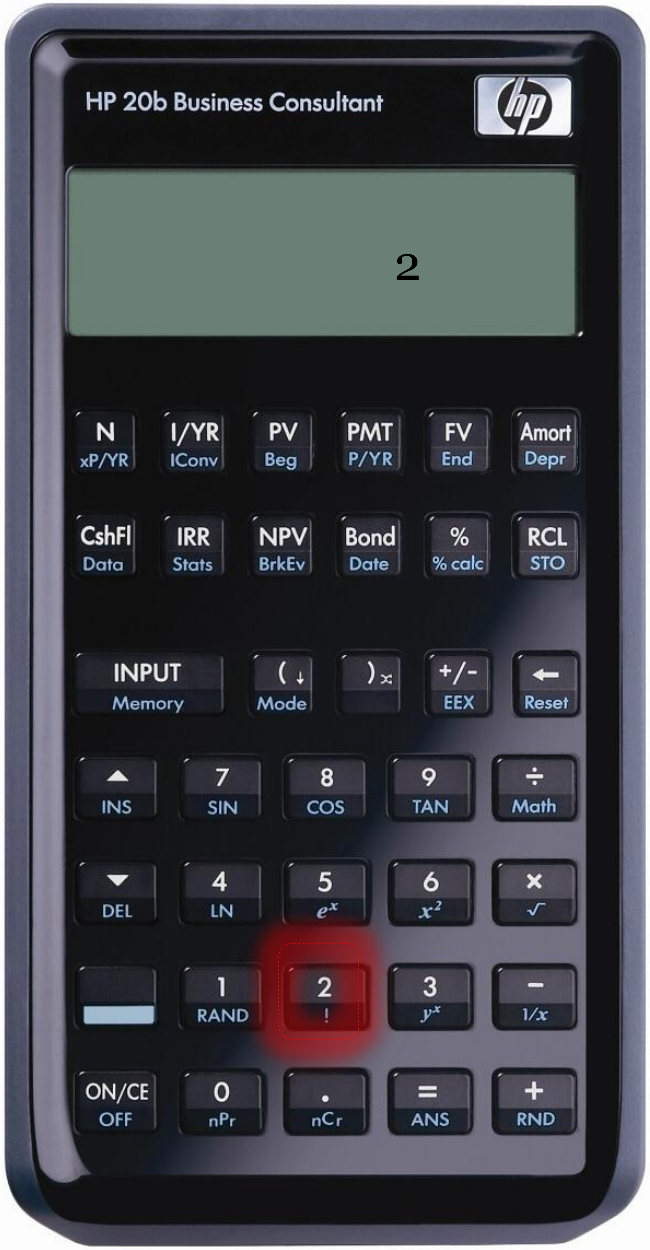
ALGEBRA

4+5 X 2



ALGEBRA

4+5 X 2

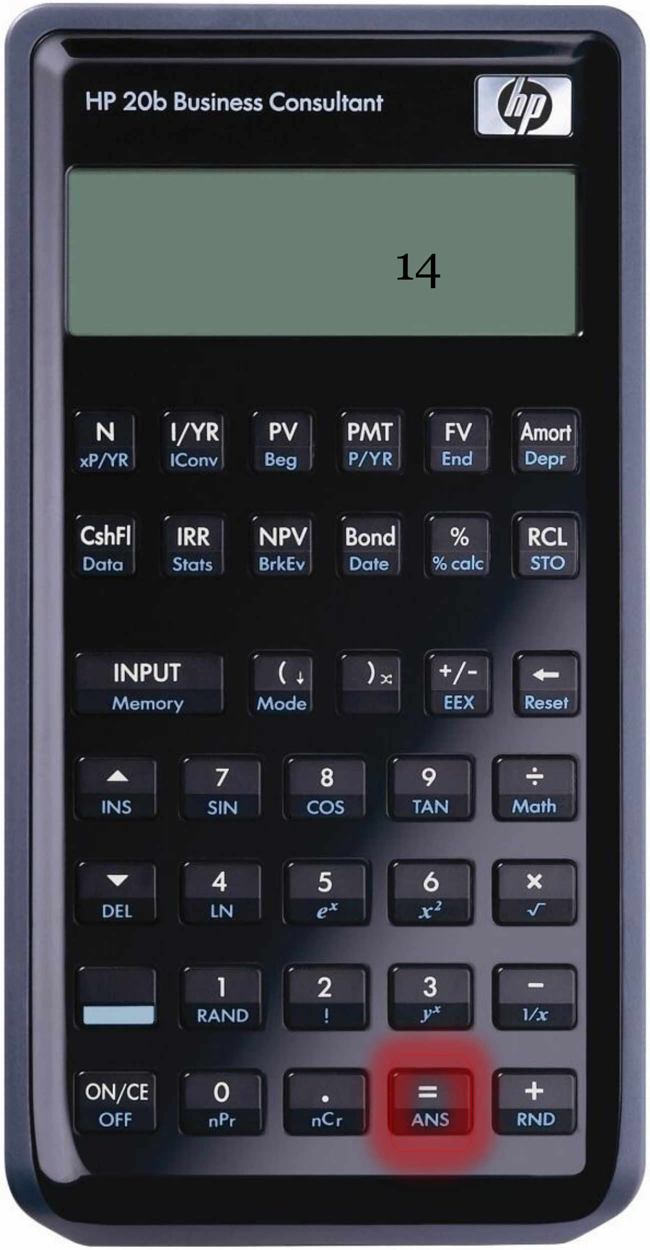


ALGEBRA

4+5 X 2

ALGEBRA

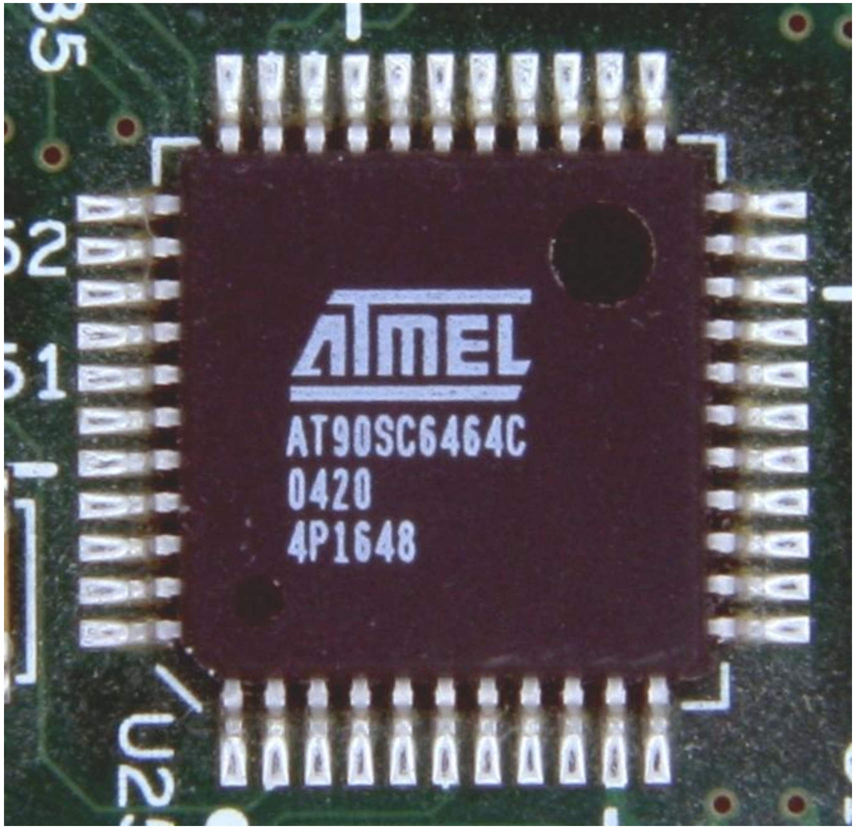
4+5 X 2



HARDWARE

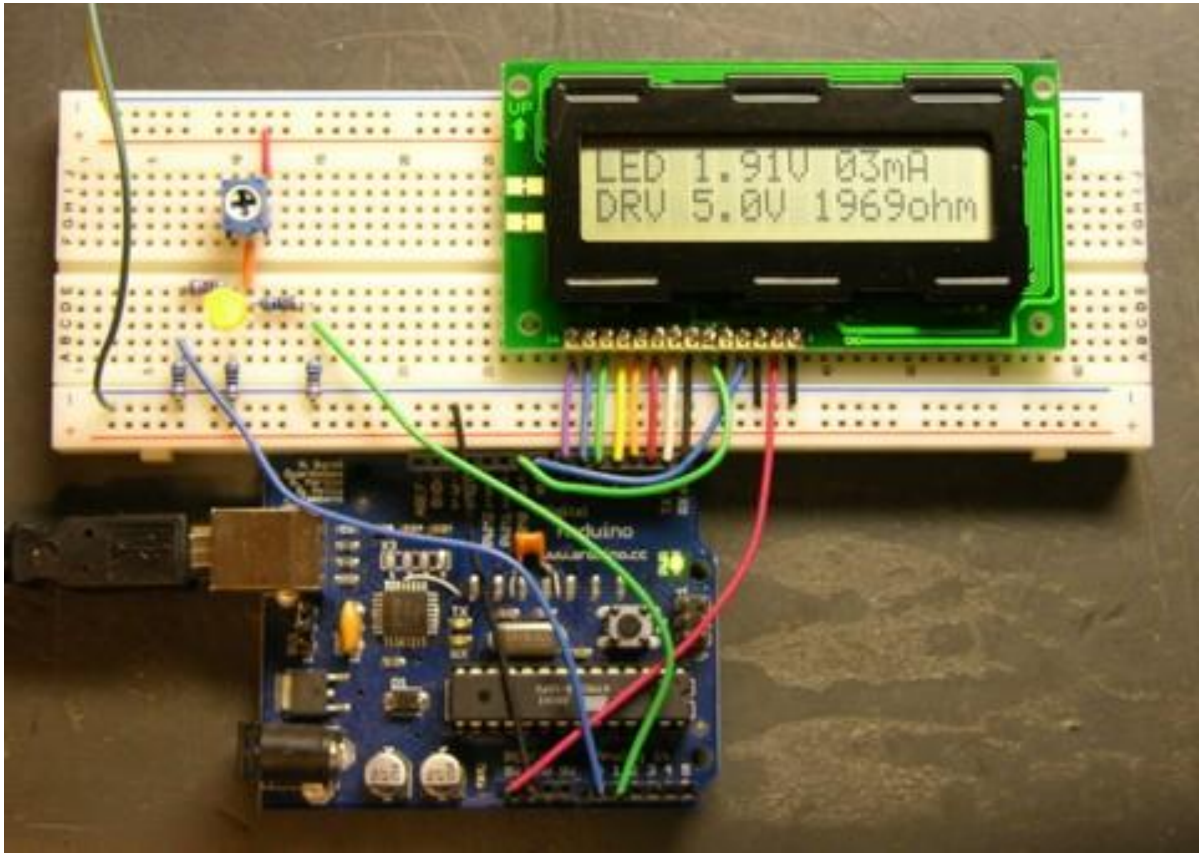
Atmel AT91SAM7L128 PROCESSOR

- "AT" is for Atmel
- "SAM" is "smart ARM core"
- 7L series of microcontrollers
  - designed for low power (hence the L)
  - Allows it to run off low voltage batteries (watch batteries)
- 128K of flash program memory

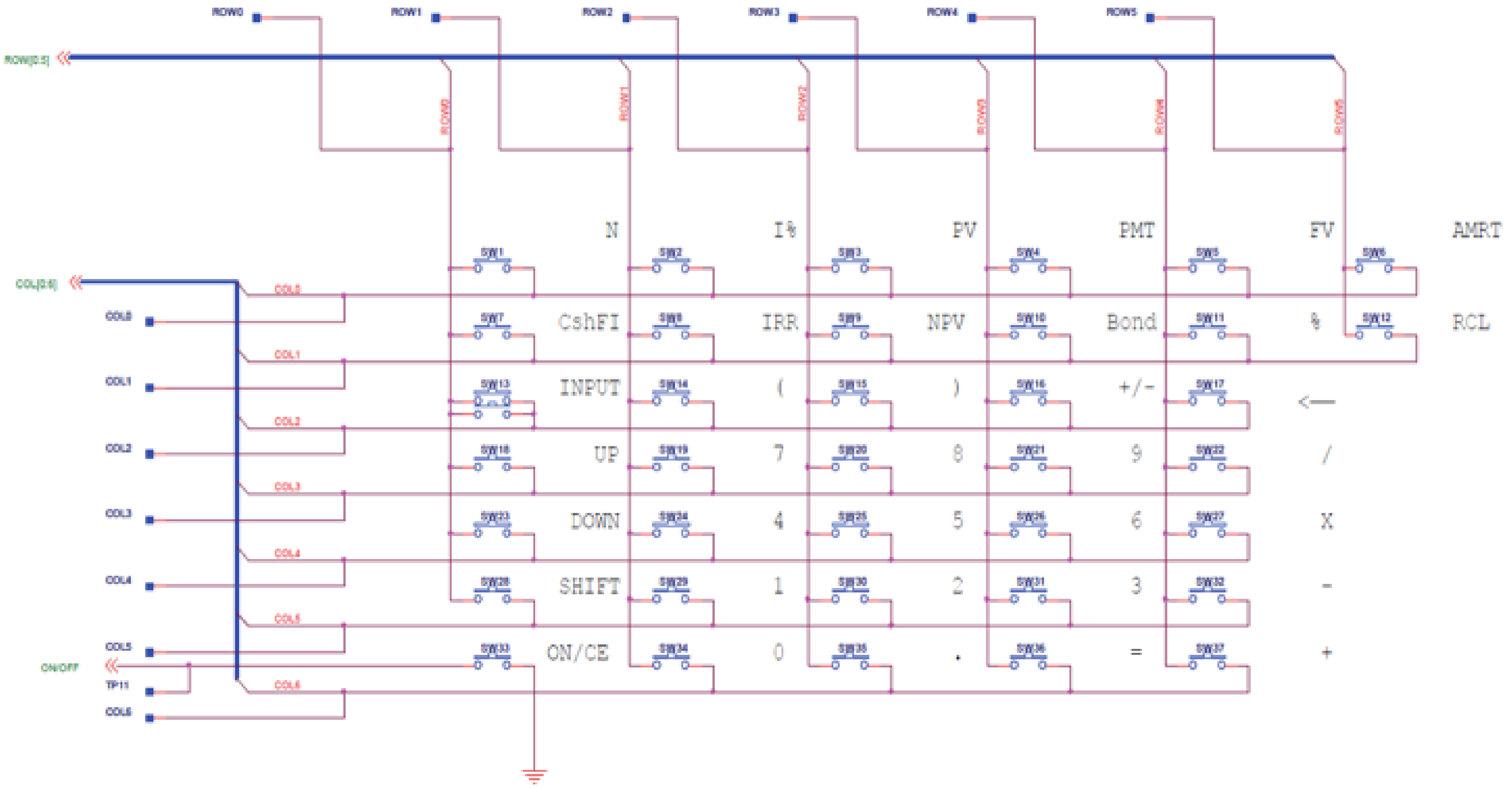


LCD DISPLAY

- 12 Digit LCD
- Large 2 line LCD display



KEYBOARD


ROW0
ROW1
ROW2
ROW3
ROW4
ROW5
ROW[0:5]
COL[0:6]
COL0
COL1
COL2
COL3
COL4
COL5
COL6
ON/OFF
TP11
N
I%
PV
PMT
FV
AMRT
CshFI
IRR
NPV
Bond
%
RCL
INPUT
(
)
+/-
<—
UP
7
8
9
/
DOWN
4
5
6
X
SHIFT
1
2
3
-
ON/CE
0
.
=
+
SW1
SW2
SW3
SW4
SW5
SW6
SW7
SW8
SW9
SW10
SW11
SW12
SW13
SW14
SW15
SW16
SW17
SW18
SW19
SW20
SW21
SW22
SW23
SW24
SW25
SW26
SW27
SW28
SW29
SW30
SW31
SW32
SW33
SW34
SW35
SW36
SW37

PROGRAMMING

LAB 1

CODE

```
int myFavoriteNumber(int x)
{
      int position = 11;
      if (x == 0) {
        lcd_put_char7(48, 11);
        return 0;
   }

   if (x < 0) {
      lcd_put_char7('-', 0);
      x = -x;
   }
   while (x != 0) {
      char d = (x%10 + 48);
      lcd_put_char7(d, position);
      x /= 10; // x = x/10
      position -= 1;
   }
   return (12-position);
}
```

EXPLANATION

- We tell the calculator to display the interger at position 11
- If the number is less than zero, display a negative sign and treat number as positive
- If the number is not 0, loop through
- **Receives numerical input from main function**
- **Displays on the right side of the screen**
- **Would later use an unsigned integer**
- **Device not yet a calculator and display the digits**

KEYBOARD.C

```
int keyboard_key () {
    int i = 0;
    int j = 0;
    for (i=0; i<7; i++) keyboard_column_high(i);

    for (i=0; i<7; i++) {
        keyboard_column_low(i);
        for (j=0; j<6; j++) {
            if (!keyboard_row_read(j)) {
                return j*10 + i;
            }
        }
        keyboard_column_high(i);
    }
    for (i=0; i<7; i++) keyboard_column_low(i);
    return -1;
}
```

- Initialize the for loop operators
- Set all colmnus to high
- Iterate through all columns
- Set this column to low
- Detect if this column is being read
- Encode x, y as a two digit number
- Set this column back to high
- Set all columns back to low
- Return -1 to indicate no input

LAB 2 AND 3

CODE

```
int main() {
…
   char A[4][4] = { {'7', '8', '9', '/'},
      {'4', '5', '6', 'X'},
      {'1', '2', '3', '-'},
      {'0', '.', '=', '+'} };

   for (;;) {
      inn = keyboard_key();
      if (inn != -1) {
         res[0] = (inn - (inn % 10))/10;
         res[1] = inn % 10;
      }
      else {
         res[0] = -1;
         res[1] = -1;
      }
```

EXPLANATION

- Forever
- This if/else block converts the two digits returned by keyboard_key into a 1x2 array, the x,y coo
- If the keyboard_key() function returns that there is no input coordinate
- **We check for the low pin values, as they indicate the button is being pressed**
- **The location on the grid is mapped onto an array**
- **The array contains the characters that we could then display**

LAB 2 AND 3

CODE

```
if (res[1]>2 && res[1]<8 && res[0]>0 &&
    res[0]<6 && len < 10) {
        if (pause == 1) {
            num*=10;
            num+=A[res[1]-3][res[0]-1] - '0';
            len = myFavoriteNumber(num);
            pause = 0;
        }
    }
    else if (res[1]==0 && res[0]==0) {
        for (j=0; j<12; j++)
            lcd_put_char7(' ', j);
        num = 0;
        myFavoriteNumber(num);
        len=0;
    }
    else if (pause == 0){
        pause = 1;
    }
  }
  ...
}
```

EXPLANATION

- If the inputs are within the number grid
- And the debounce is disabled
- Enable the debounce
- If the ‘reset’ button is struck
- Clear the screen
- Redisplay 0
- Disable the debounce
- **Using a makeshift reset button, would later employ On-Clr Button**
- **Still not a calculator**

LAB 4

- Parallel of operations makes this method easily condensible
- Stack depth is semi-arbitrary, but it was set to 5
- Device is now a calculator

```
for (;;) {
    keyboard_get_entry(&beta);
    if (beta.operation == 'q') {
        opp = &op[0];
    }
    else if (beta.operation == '\r') {
        *opp = beta.number;
        opp++;
        while(keyboard_key() != -1) {
            continue;
        }
    }
    else if (beta.operation == '+' || beta.operation == '-' ||
beta.operation == '*') {
        if (beta.newNum == 1)
            *opp = beta.number;
        else
            opp--;

        if (opp > &op[0]) {
            if (beta.operation == '+')
                *(opp-1) = *(opp-1) + *opp;
            else if (beta.operation == '-')
                *(opp-1) = *(opp-1) - *opp;
            else if (beta.operation == '*')
                *(opp-1) = *(opp-1) * *opp;
            myFavoriteNumber(*(opp-1) < 0 ? -*(opp-1) : *(opp-1),
*(opp-1) < 0);
        }
        else {
            lcd_put_char7('r', 1);
            if (beta.newNum == 0)
                opp++;
        }

        while(keyboard_key() != -1) {
            continue;
        }
    }
}
```

LAB 5

- Addition, subtraction, and multiplication are consistent with the order of operations
- Didn't have time to optimize code properly, or develop parenthesis
- A functioning calculator in the traditional sense

```
for (;;) {
    keyboard_get_entry(&beta);
    if (beta.operation == 'q') {
      opp = &op[0];
      xSign = 1;
      pHold = 1;
    }
    else if (beta.operation == '+' || beta.operation == '-') {
      if (beta.newNum == 1) {
        *opp = beta.number;
        if (opp == &op[0]) {
          opp++;
        }
        else if (opp == &op[1]) {
          *(opp-1) += *opp * xSign;
          myFavoriteNumber(*(opp-1) < 0 ? -*(opp-1) : *(opp-
   1), *(opp-1) < 0);
        }
        xSign = (beta.operation == '-' ? -1 : 1);
      }
      while(keyboard_key() != -1) {
        continue;
      }
    }
    else if (beta.operation == '*') {
      do {
```

```
if (beta.newNum == 1) {
            pHold *= beta.number;
            myFavoriteNumber(pHold < 0 ?
-pHold : pHold, pHold < 0);
            keyboard_get_entry(&beta);

            while(keyboard_key() != -1) {
              continue;
            }
          }
```

LAB 5

- Addition, subtraction, and multiplication are consistent with the order of operations
- Didn't have time to optimize code properly, or develop parenthesis
- A functioning calculator in the traditional sense

```
else {
          keyboard_get_entry(&beta);
        }
    } while(beta.operation == '*');

    if (beta.operation == '+' || beta.operation == '-') xSign =
  (beta.operation == '-' ? -1 : 1);

    if (opp == &op[0]) {
        *opp = pHold * beta.number;

        if (beta.operation == '=') {
          myFavoriteNumber(*opp < 0 ? -*opp : *opp, *opp <
  0);
          opp = &op[0];
        }

        else {
          opp++;
        }
    }
    else if (opp == &op[1]) {
        *(opp-1) += pHold * beta.number * xSign;
        myFavoriteNumber(*(opp-1) < 0 ? -*(opp-1) : *(opp-1),
  *(opp-1) < 0);
    }
```

```
pHold = 1;

          while(keyboard_key() !=
-1) {
             continue;
          }
       }
```

LAB 5

- Addition, subtraction, and multiplication are consistent with the order of operations
- Didn't have time to optimize code properly, or develop parenthesis
- A functioning calculator in the traditional sense

```
else if (beta.operation == '=') {
      if (beta.newNum == 1) {
        *opp = beta.number;
        if (opp == &op[1]) {
          if (tOp == '-' || tOp == '+') {
            *(opp-1) += *opp * xSign;
            myFavoriteNumber(*(opp-1) < 0 ? -*(opp-1) : *(opp-1), *(opp-1) < 0);
            opp = &op[0];
          }
        }
      }

      while(keyboard_key() != -1) {
        continue;
      }
    }
    tOp = beta.operation;
  }
```

SOCIAL IMPLICATIONS

REFLECTION

LESSON LEARNED

- Plan ahead
- Be organized

CRITICISM

- Assumed knowledge of C makes it hard for those without solid programming knowledge to participate
- More time should be sectioned off to teach C

FINAL THOUGHTS