```
| calcul du n-ième nombre premier.
.include beta


init :
        BR(start,r31)
n:
        LONG(1000)
result:
        LONG(0)
start:
        CMOVE(stack,SP)
        LD(r31,n,r0)
        PUSH(r0)                  | passage de l'argument
        BR(prime,LP)
        ST(r0,result,r31)         | store the result.
end:
        BR(end,r31)               | infinite sleep.

| calcule le n-ième nombre premier (2:1er nbre premier).
| n est passé en argument sur la pile.
| Le nombre premier de numéro n est retourné dans r0.
prime:
        PUSH(LP)                  | sauver LP n'est pas necessaire ici...
        PUSH(BP)

| acces au cadre de pile par BP.
        MOVE(SP,BP)

| r1..r5 vont etre utilises...
        PUSH(r1)
        PUSH(r2)
        PUSH(r3)
        PUSH(r4)
        PUSH(r5)

| recuperation de l'argument dans r0
        LD(BP,-12,r0)

| registres pour variables de travail
| r1: p nombre teste courant
| r2: k compteur de nombres
| r3: d diviseur courant
| r4: registre temporaire tmp
| r5: registre temporaire tmp1

p=r1
k=r2
d=r3
tmp=r4
tmp1=r5

| si n=1 on saute en "one"
        SUBC(r0,1,tmp)
        BEQ(tmp,one,r31)
| n > 1
| initialisations:
        CMOVE(1,p)                | p <- 1
        MOVE(p,k)                 | k <- 1
main_loop:
        SUB(k,r0,tmp)             | if k=n then...
        BEQ(tmp,exit,r31)         | goto exit

        ADDC(p,2,p)               | p <- p+2     parcours des nombres pairs.
        CMOVE(2,d)                | d <- 2
```

```
next_div:
        MUL(d,d,tmp)               | tmp <- d*d
        CMPLE(tmp,p,tmp)           | tmp <- d*d <= p

        DIV(p,d,tmp1)              | tmp1 <- p mod d
        MUL(tmp1,d,tmp1)           | ...
        SUB(p,tmp1,tmp1)           | ...

        BEQ(tmp1,tmp1OK,r31)       | if(tmp1 <> 0) tmp1 <- 1
        CMOVE(1,tmp1)

tmp1OK:
        AND(tmp,tmp1,tmp)          | tmp <- tmp AND tmp1
        BEQ(tmp,next_prime,r31)    | while(d*d <= p and p mod d <> 0) d <- d+1

        ADDC(d,1,d)                | d <- d + 1
        BR(next_div,r31)

next_prime:
        MUL(d,d,tmp)               | tmp <- d*d
        CMPLE(tmp,p,tmp)           | tmp <- d*d <= p
        BNE(tmp,main_loop,r31)     | if(d*d > p) k <- k+1 else goto main_loop
        ADDC(k,1,k)                | ...
        BR(main_loop,r31)          | goto main_loop

| rendre 2, le premier nombre premier, dans r0
one:
        CMOVE(2,r0)
        BR(exit2,r31)
exit:
        MOVE(p,r0)                 | si fini, resultat dans r0...
exit2:
        POP(r5)
        POP(r4)
        POP(r3)
        POP(r2)
        POP(r1)

        POP(BP)
        POP(LP)

        JMP(LP,r31)
stack:
```