Instrukcje BSR i RTS – implementacja w mikrokodzie

Zaimplemetować instrukcje skoku i powrotu z podprogramu:

- BSR – (Branch to SubRoutine) jest skokiem względnym (tj. PC=PC+offset) do adresu podanego jako etykieta, z odłożeniem na stos rejestru PC
- RTS – (ReTurn from Subroutine) jest skokiem absolutnym (tj. PC=address) do adresu odczytanego ze stosu

- W obu przypadkach wskaźnikiem stosu jest rejestr Rx

Implementacje wykonaj w projekcie lab3 z poprzednich zajęć z PUSH/PULL

1 **BSR Rx,label**

→ deklaracja: **BSR I-type r1,j**

1) PC → stack(Rx)
2) PC + signext(IR) → PC

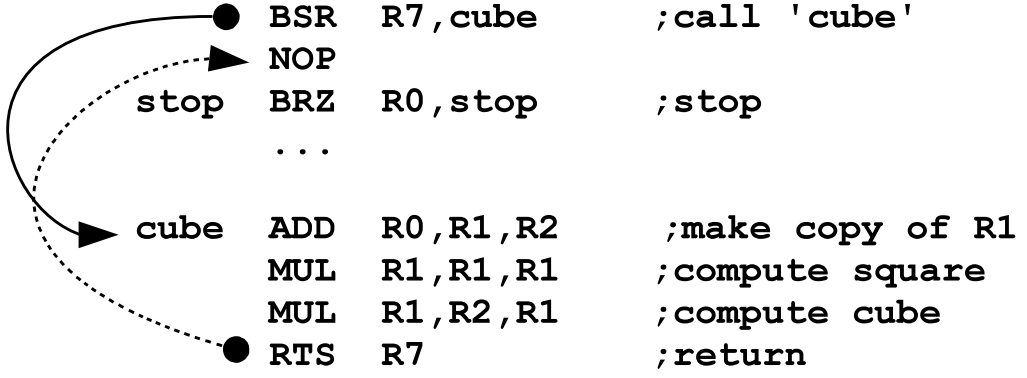
2 **RTS Rx**

→ deklaracja: **RTS R-type r1**

1) stack(Rx) → PC

Pamietaj, że instrukcje BSR i RTS wykonują operacje na stosie w podobny sposób do PUSH i PULL, czyli modyfikują wskaźnik stosu.

Testowanie BSR i RTS



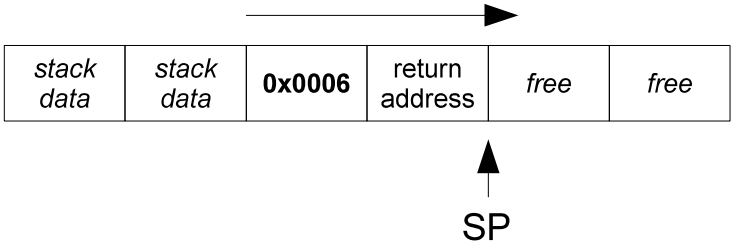
Obliczanie silnii z wykorzystaniem rekursji

3 $n! = n \cdot (n-1)! \rightarrow fact(n) = n \cdot fact(n-1)$

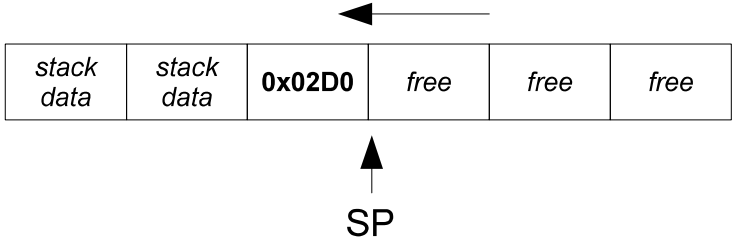
Funkcja `fact()` pobiera jeden parametr (32-bit int) ze stosu i zwraca wynik (32-bit int) na stosie w miejscu pobranego parametru

```
// C implementation
int fact(int n)
{
    if n==1 return n
    else return n*fact(n-1);
}
```

zawartość stosu po wykonaniu pierwszego BSR



zawartość stosu po wykonaniu ostatniego RTS



Struktura programu:

```
        ADDI R0, 0x????, R7    ;init SP
        ADDI R0, 0x0006, R1    ;set n=6
        PUSH R7, R1            ;pass n to fact
        BSR  R7, fact          ;call fact
        PULL R7, R1            ;get result n!
        NOP
stop    BRZ  R0, stop
        ...

fact    LDW  R2, offset(R7)    ;read parameter
        ...
        ...
        ...
        STW  R2, offset(R7)    ;return result
return  RTS  R7
        NOP
```

Wartość przesunięcia `offset` zależy od implementacji stosu: +4 lub +8