Вызов подпрограмм
Передача аргументов

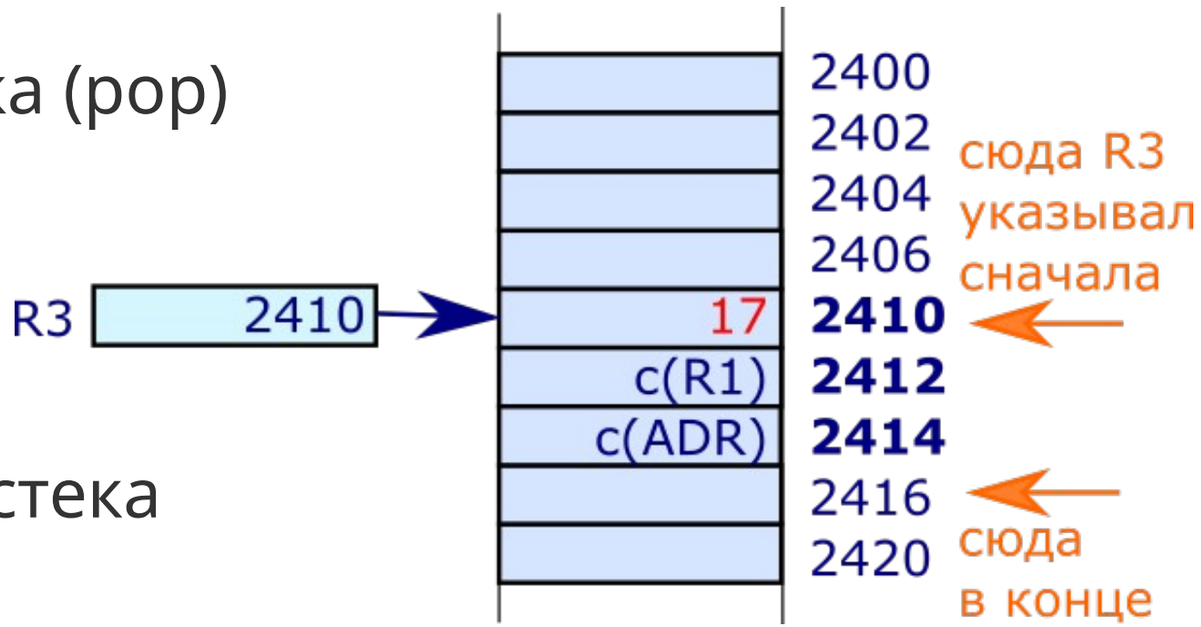
Курсовой проект "Эмулятор PDP-11"
Занятие 8

Задача урока

- Напечатать Hello, world! с помощью функций
  - Реализация команд jsr, rts
- Понять как можно передать аргументы в функцию

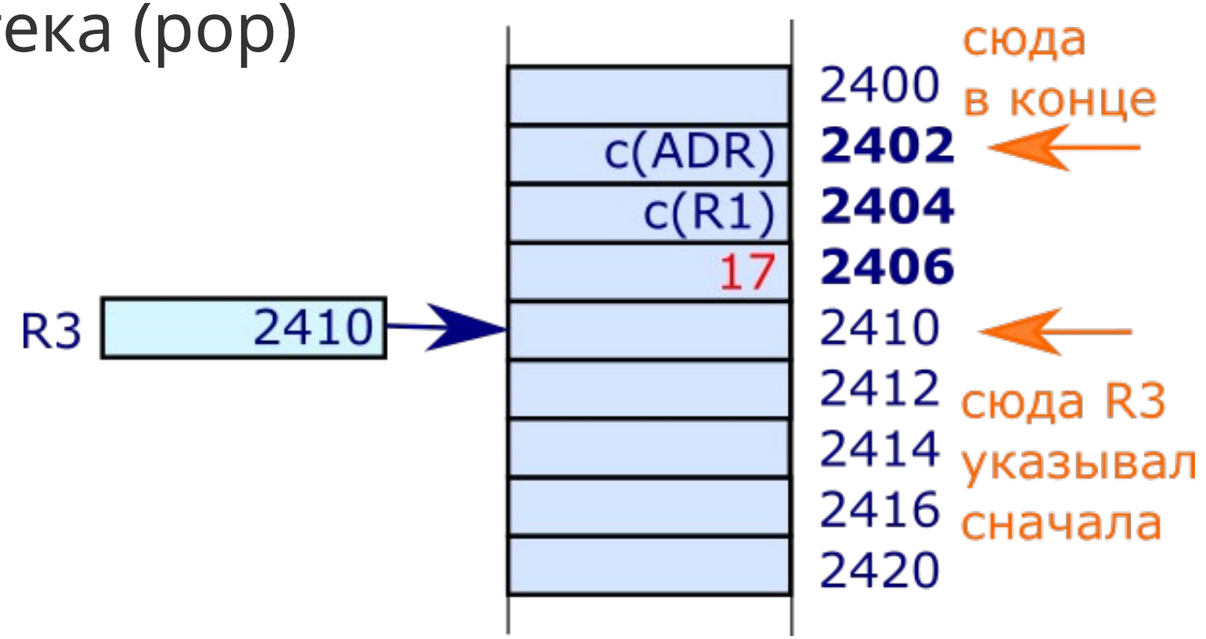
Стек

- Пусть сначала в регистре R3 лежит число 002410
- кладем в стек (push)
  mov #17, (R3)+
  mov R1, (R3)+
  mov ADR, (R#)+
- достаем из стека (pop)
  mov -(R3), R4
- R3 указывает
  на первую
  пустую ячейку стека



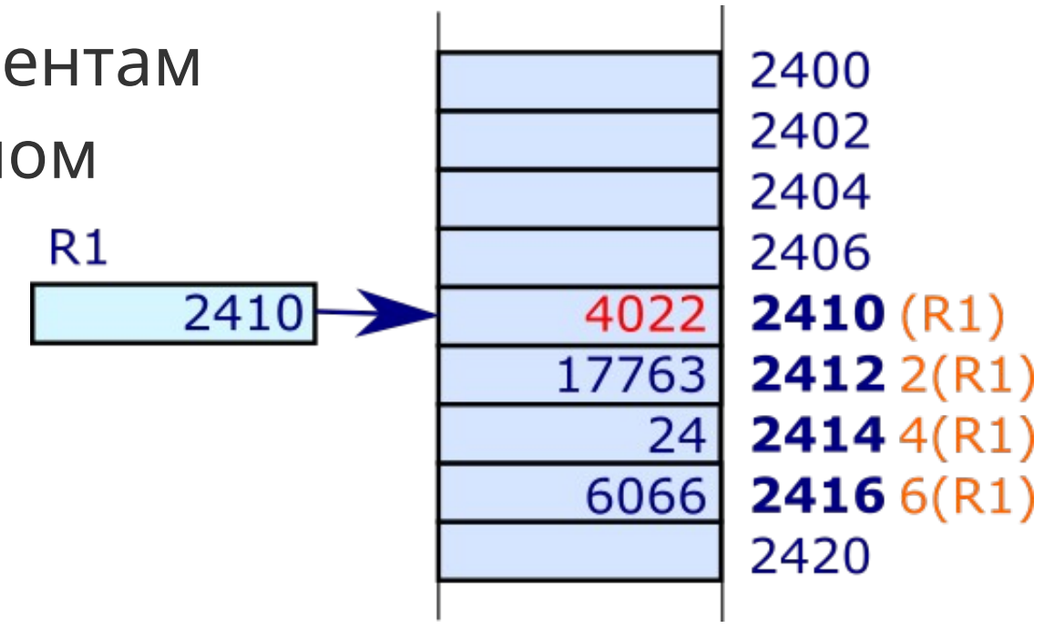
Стек

- Пусть сначала в регистре R3 лежит число 002410
- кладем в стек (push)
  mov #17, -(R3)
  mov R1, -(R3)
  mov ADR, -(R3)
- достаем из стека (pop)
  mov (R3)+, R4
- R3 указывает на вершину стека



Доступ внутрь стека

- R1 — указатель на вершину стека
- clr R5 ; R5 = 0
  add 6(R1), R5 ; R5 += 6066
  add (R1), R5 ; R5 += 4022
  add 2(R1), R5 ; R5 += 17763
- доступ к элементам в произвольном порядке



Программный стек

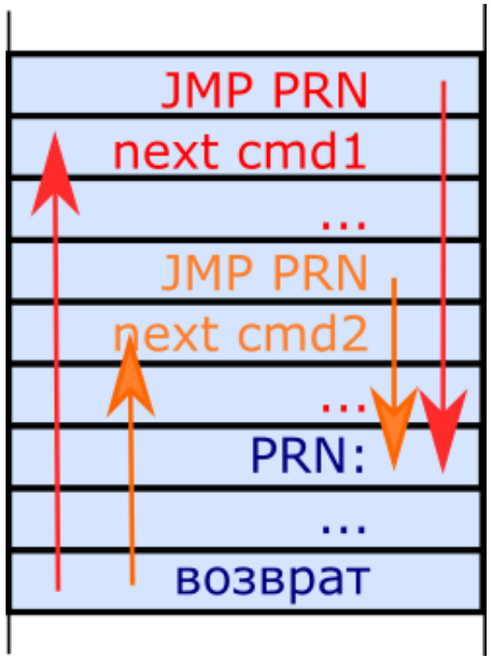
- В большие адреса
  2000 **B: .BLKW 10**
  стек на 10 слов по адресу B
- В меньшие адреса (от программы в другую сторону)
  1000 **START: mov #START, R4**
  R4 используется как указатель стека
- Проблема — смешанный доступ
  нужно самим следить за четностью адресов
  1002 **mov** #17, -(R4)
  1006 **movb** #'z, -(R4)

Аппаратный стек SP = R6

- Стек на основе R6 называют аппаратным,
  потому что существуют аппаратные инструкции,
  которые его используют
- Моды адресации **-(R6)** и **(R6)+** имеют особые правила
  всегда на 2.
- SP — stack pointer

Подпрограммы (subroutines)

- Перейти к подпрограмме по адресу начала подпрограммы
  JMP PRN
- Адрес возврата после выполнения подпрограммы. Его нужно где-то хранить
- Аргументы подпрограммы
- Возвращаемое значение



Храним адрес возврата в регистре R5

024614 010705 mov PC, R5

024616 000167 jmp PRINT

024620 021622

024622 следующая команда

…

046444 PRINT: начало процедуры

…

046552 010507 mov R5, PC

вернулись не туда: 024616 вместо 024622

Поправка перед возвратом

024614 010705 mov PC, R5

024616 000167 jmp PRINT

024620 021622

024622 следующая команда

...

046444 PRINT: начало процедуры

...

046552 062705 add #4, R5

046554

046556 010507 mov R5, PC

Поправка перед вызовом

024614 010705 mov PC, R5

024616 062705 add #10, R5

024620 000010

024622 000167 jmp PRINT

024624 021622

024626 следующая команда

...

046444 PRINT: начало процедуры

...

046552 010507 mov R5, PC

Минусы хранения в регистре

- Регистров мало, нужны для вычислений
- Вызов из одной процедуры другой — сложно.
- Рекурсия — невозможно
- Старое значение регистра R5 может понадобиться, а мы его уничтожили
- Идеи:
  - хранить адрес возврата в стеке по SP
  - хранить старое значение регистра в стеке по SP

Храним адрес возврата в стеке

```
024612 010505 mov R5, -(SP)
024614 010705 mov PC, R5
024616 062705 add #10, R5
024620 000010
024622 000167 jmp PRINT
024624 021622
024626 mov (SP)+, R5
...
046444 PRINT: начало процедуры
046552 010507 mov R5, PC
```

JSR и RTS

024612

024614 004567 JSR R5, PRINT

024616 021624

024620 (следующая)

...

046444 PRINT: начало процедуры

...

046552 000205 RTS R5

Вызов подпрограммы и возврат из нее

- **JSR Rn, adr**
  - jump to subroutine
  - Сохранить содержимое регистра Rn в стек
  - Поместить (правильный) адрес возврата в стек
  - Передать управление подпрограмме по adr
- **RTS Rn**
  - return from subroutine
  - возвращаемся из подпрограммы
  - восстанавливаем значение регистра Rn

SUB1 вызывает SUB2

```
024612 010505 mov #12, R2
024614 004267 JSR R2, SUBA
024616 ???????
024620 ...
046444 SUBA: ...
046450 004267 JSR R2, SUBB
046452 ?????? ...
046464 000202 RTS R2
046466 SUBB: ...

...
046572 000202 RTS R2
```

Передача аргументов

- Как мы писали для putc,
  можно в определенный регистр

```
    mov  #'h, R0
    jsr     R2, putc
    halt
putc:
    tstb   ostat
    bpl    putc
    movb R0, odata
```

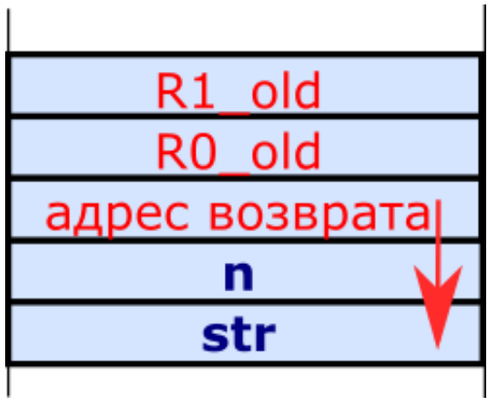
- Что делать, если аргументов много?
- Пишем puts — какой регистр использовать для передачи адреса начала строки

Передача фреймом (фортран)

```
JSR R2, putsn ; Положим аргументы сразу за вызовом JSR
    .WORD str     ; адрес начала строки
    .WORD n       ; сколько символов печатать
    сюда вернуться
putsn:
    mov R0, -(SP)    ; сохранили R0
    mov R1, -(SP)    ; сохранили R1
    mov (R2)+, R0    ; слово после jsr
    mov (R2)+, R1    ; следующий аргумент
    ...
    mov (SP)+, R1    ; восстанавливаем R1
    mov (SP)+, R0    ; восстанавливаем R0
    rts R2
```

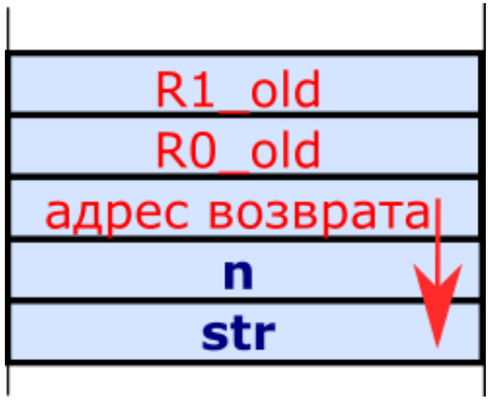
Передача через стек (паскаль)

```
mov #str, -(SP)    ;    положим аргументы перед вызвом в стек
mov #n, -(SP);
JSR PC, putsn ;
сюда вернуться, адрес хранится в стеке
putsn:
mov R0, -(SP)  ; сохранили R0
mov R1, -(SP)  ; сохранили R1
mov 10(SP), R0    ; str
mov 6(SP), R1 ; n
...
mov (SP)+, R1 ; восстанавливаем R1
mov (SP)+, R0 ; восстанавливаем R0
mov (SP)+, 2(SP) ; передвигаем вершину стека
add #2, SP       ; tst (sp)+
rts PC
```



Передача через стек (Си)

```
mov #str, -(SP)    ;    положим аргументы перед вызвом в стек
mov #n, -(SP);
JSR PC, putsn ;
add #4, SP          ; сюда вернуться, адрес хранится в стеке
...
putsn:
mov R0, -(SP)  ; сохранили R0
mov R1, -(SP)  ; сохранили R1
mov 10(SP), R0    ; str
mov 6(SP), R1 ; n
...
mov (SP)+, R1 ; восстанавливаем R1
mov (SP)+, R0 ; восстанавливаем R0
rts PC
```



JSR Rn, adr

- R0.. R5
  - TEMP = адрес перехода
  - push(SP) = c (Rn)
  - Rn = c (PC)
  - PC = TEMP
- R7
  - TEMP = адрес перехода
  - push(SP) = c (R7)
  - PC = c (PC) холостой ход
  - PC = TEMP

RTS Rn

- R0..R5
  - PC = c (Rn); возвращаемся из программы
  - Rn = pop(SP) восстанавливаем значение Rn
- R7
  - PC = c (R7) ; холостой ход
  - PC = pop(SP) ; возвращаемся из программы

Задача урока

- Реализовать моду 6 (если ее еще не было)
- Реализовать jsr, rts
- Тесты:
  - печать hello, world в виде функции
  - рекурсивный вызов функции