UMP OPEN COURSEWARE

Universiti Malaysia PAHANG
Engineering • Technology • Creativity

Chapter 8

Subroutine

Expected Outcomes

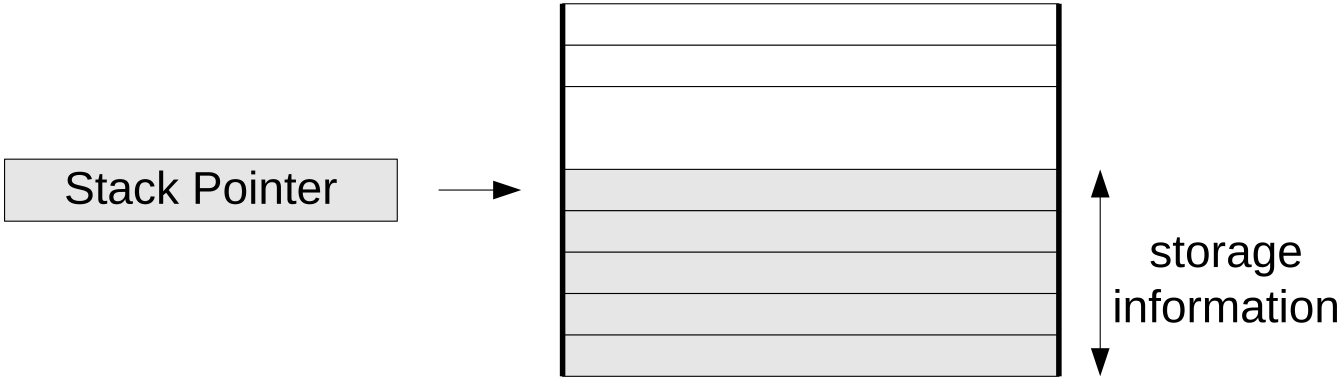
- Describe and apply the stack for data storage
- Describe the process of subroutine in any programs
- Develop a subroutine and code
- Interpret subroutine process in the stack
- Write and calculate a delay subroutine

Stack

- Stack is a special area in memory and normally it is used to keep track of and store CPU register information during execution
- Most stack uses LIFO concept
- It has a **stack pointer (SP)** to indicate where to push or pull data and A7 is used for this purpose
- To store data, CPU pushes it onto the stack and then decrements the SP
- To recover data from the stack, it increments the SP and then pulls the data
- The stack grows toward low memory addresses

Stack Pointer

- Stack must be located in RAM location and normally place above program and data
- **Stack pointer** must be initialized at the beginning of the program



Push & Pull Instruction

- ARI with pre-decrement and ARI with post-increment are required to perform the push and pull operation
- Push:

```
MOVE.s Source,-(SP)
```

- Pull:

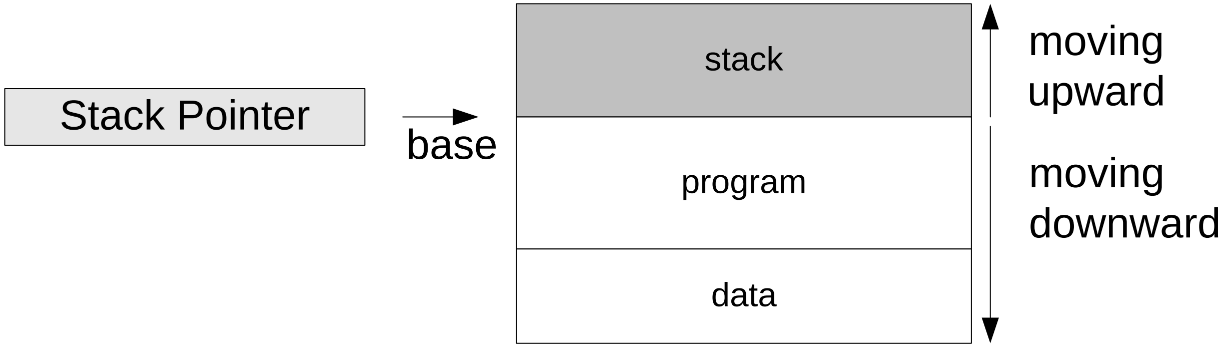
```
MOVE.s (SP)+,destination
```

Initialize Stack Pointer

- To initialize stack pointer

```
MOVEA.L #BASE,SP
```

where BASE is the end address of a stack



Subroutine

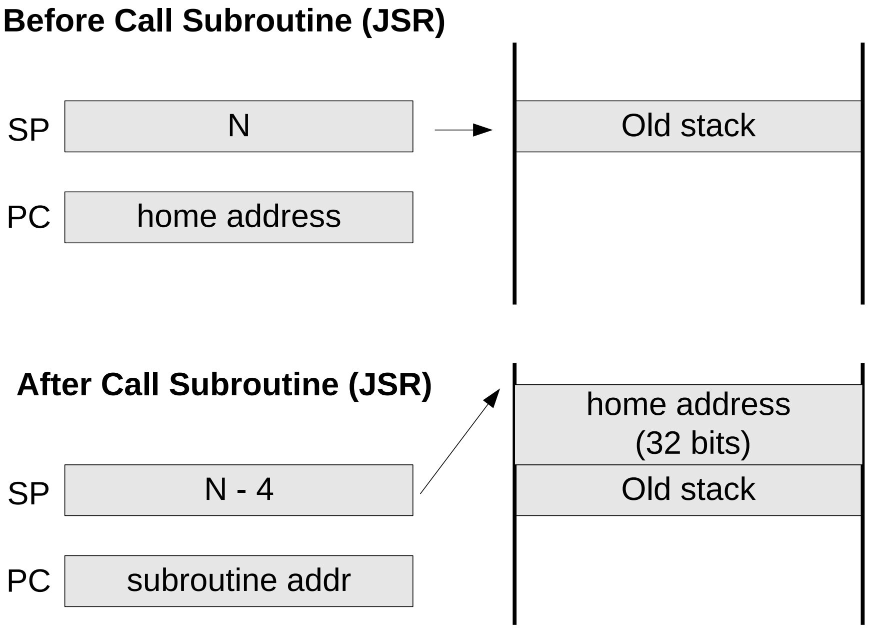
- Subroutine is a section of a program that may be used one or more times
- With subroutine, the program is much simpler, short, efficient and more understandable
- The main program calls subroutine to perform certain steps using the instruction **JSR** (jump to subroutine) or **BSR** (branch to subroutine)
- It executes the subroutines until the instruction **RTS** (return from subroutine)
- It returns to main program and continues at the instruction following instruction **JSR** or **BSR**

Subroutine

```
*   Main program
START   ……
        JSR SUB ; call routine
        ……
        BSR SUB ; call  routine
        ……
*   Subroutine
SUB     ……
        RTS
```

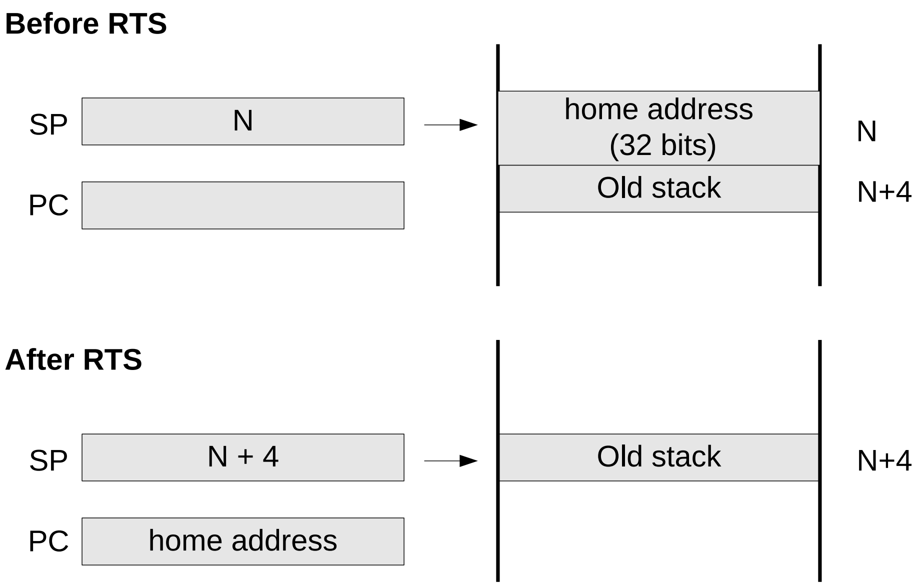
JSR Operation

- In order to return to main program, the current PC must be stored in the stack



RTS Operation

- In order to return to main program, the return address of main program must be placed back to the PC.
- RTS will ensure the procedure is followed



Before RTS

SP N

PC

home address (32 bits) N

Old stack N+4

After RTS

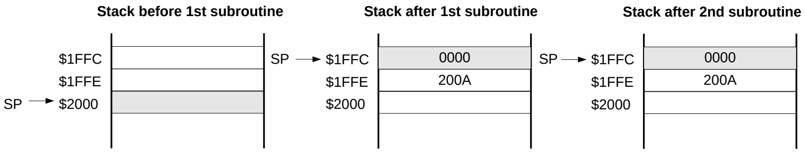
SP N + 4

PC home address

Old stack N+4

JSR & RTS Operation

```
                    ORG $2000
             STACK EQU *
2000         START MOVEA #STACK,SP
2004               NOP
2006               JSR SUB
200A         RET1  NOP
200C               JSR SUB
2010         RET2  NOP
2012               BRA *
2014         SUB   NOP
2016               RTS
```



Stack before 1st subroutine

$1FFC
$1FFE
SP → $2000

Stack after 1st subroutine

SP → $1FFC 0000
$1FFE 200A
$2000

Stack after 2nd subroutine

SP → $1FFC 0000
$1FFE 200A
$2000

Protecting Registers

- When subroutine is executed, the content of registers may alter if the registers are used in the subroutine
- Thus, the register need to be stored in stack
- Following is one way to secure the content of registers (In this case, register `D1-D2` and `A3,A6`)

```
MOVE.L   D1, -(SP)
MOVE.L   D2, -(SP)
MOVE.L   A3, -(SP)
MOVE.L   A6, -(SP)
….content of
   subroutine….
MOVE.L   (SP)+, A6
MOVE.L   (SP)+, A3
MOVE.L   (SP)+, D2
MOVE.L   (SP)+, D1
RTS
```

MOVEM Instruction

- Another alternative way to store the content of register is using the **MOVEM** (move multiple registers) instruction
- Syntax

```
MOVEM.s <list registers>,-SP
```

```
MOVEM.s SP+, <list registers>
```

- Following is an example to store register D1-D3 and A3, A4, A6

```
MOVEM.L  D1-D3/A3-A4/A6, -(SP)
…content of subroutine…
MOVEM.L  (SP)+, D1-D3/A3-A4/A6
RTS
```

Macros vs Subroutines

- Both permits a group of instruction to be defined in a single entity with a unique given label or name called up when needed
- A subroutine is called by `BSR` or `JSR` instructions, while macro is called by simply its name
- Macros are not substitute for subroutines
- Support for subroutines is provided by CPU as it is part of instruction set, while support for macros is part of the assembler

Self-Test

- **Exercise**

  If SP=$0040000C and PC=$00400500, what is the value of SP when JSR $00400600 is executed ?

- **Exercise:**

  Calculate the value of SP if the following program is executed

```
      ORG  $4000
START MOVEA #$2000,SP
      MOVEM.L D0-D2/A0/A4-A6,-(SP)
```