Lecture 6: Subroutines I

- **Introduction to subroutines**
- **The stack**
- **BRS/JSR and RTS instructions**
- **An example: compute the statistical average**
- **Parameter passing to subroutines**
  - Passing by value
  - Passing by reference

Introduction to subroutines

- **What is a subroutine?**
  - A subroutine is a coherent sequence of instructions that carries out a well-defined function
  - Conceptually, a subroutine is similar to a function call in a high-level language
- **Why subroutines?**
  - The same sequence of instructions can be used many times without the need to rewrite them over and over
  - Subroutines make programs easier to write (in a top-down fashion) and maintain
- **Examples of subroutines**
  - Print a message to the display, compute an average value, initialize an I/O port, any more?

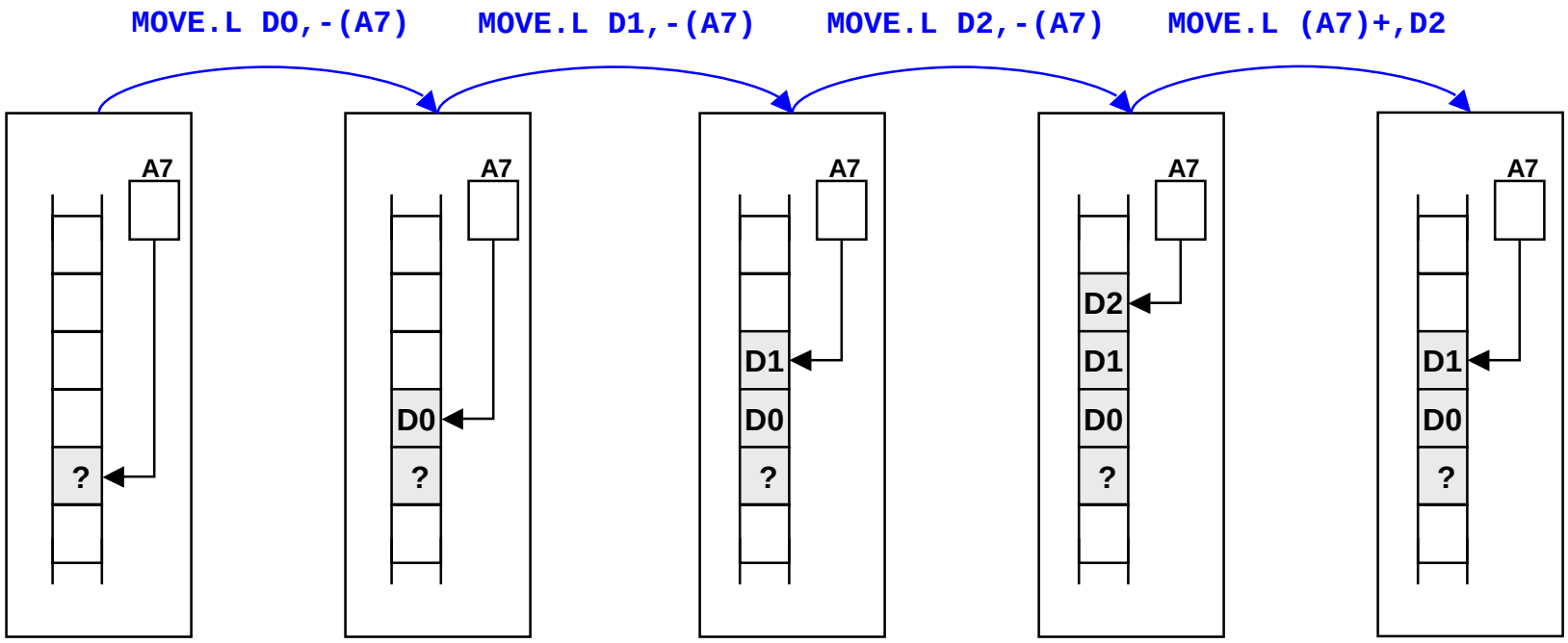
Subroutines made easy

- **When a program calls a subroutine**
  - The address (N) of the next instruction in the program is saved in a special memory location called the **STACK**
  - The PC is loaded with the starting address of the subroutine
  - The CPU performs another "fetch-execute" cycle… *this time at the first instruction of the subroutine!*
- **When a subroutine is done**
  - The old address (N) is recovered from the **STACK**
  - The program counter is loaded with (N)
  - The CPU performs another "fetch-execute" cycle… *this time at the next instruction in the program after the subroutine call!*

The stack

- **The stack is a special area in memory reserved for reading and writing special data items**
  - The stack is a LIFO structure (LAST IN, FIRST OUT) since the last item pushed is always the first item popped
  - The address of the last item pushed on the stack is stored in A7, also referred to as the Stack Pointer (SP)
- **Data is pushed (saved) on the stack with `MOVE.L`**
  - **`MOVE.L D0,-(A7)`**
- **Data is popped off (recovered from) the stack with `MOVE.L`**
  - **`MOVE.L (A7)+,D0`**
- **Always move long-words to the stack!!!**

The stack (an example)



BSR/JSR instruction: calling a subroutine

- **BSR/JSR used to Branch(Jump) to a SubRoutine**
- **BSR/JSR performs three operations**
  - Decrement the stack pointer
  - Push the program counter on the stack
  - Load the program counter with the target address
- **For example, in RTL "BSR AVG" is equivalent to**

```
[A7]       ←[A7]-4
[M([A7])]  ←[PC]
[PC]       ←AVG
```

RTS instruction: returning from a subroutine

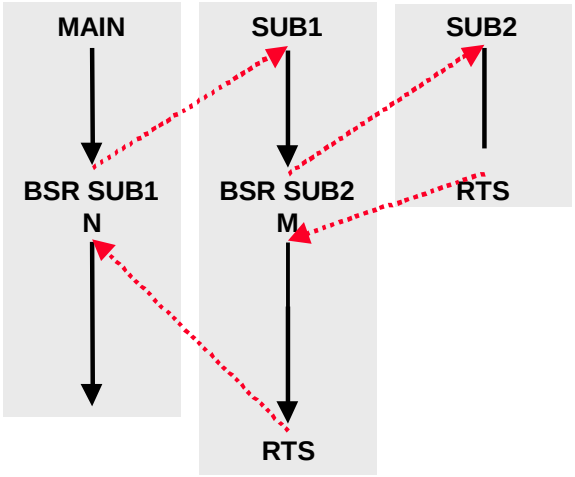
- **RTS used to ReTurn from a Subroutine**
- **RTS performs two operations**
  - Pull the return address from the stack
  - Post-increment the stack pointer
- **For example, in RTL "RTS" is equivalent to**

  [PC]←[M([A7])]

  [A7]←[A7]+4

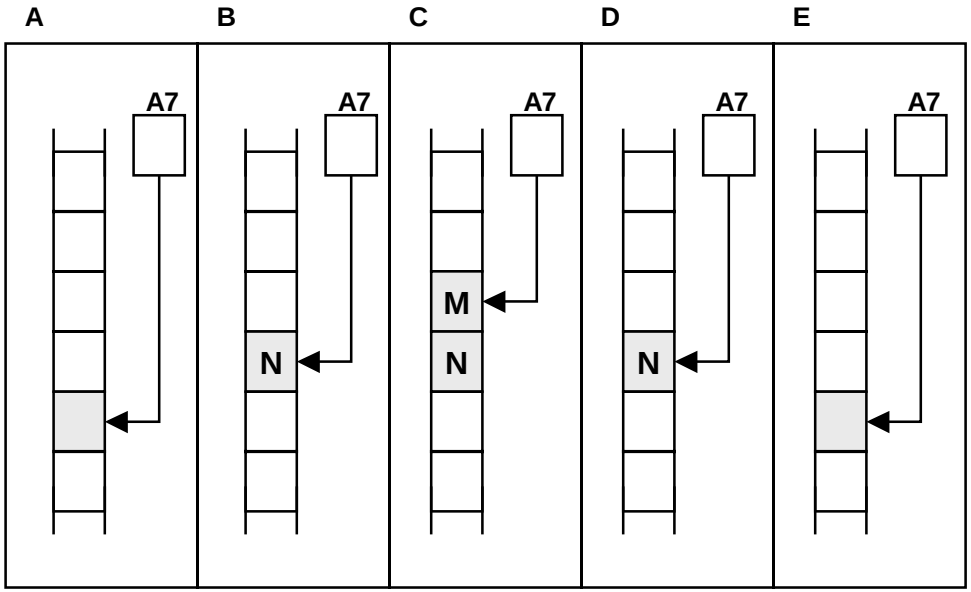
Subroutines and the stack

```
MAIN    ...
        BSR     SUB1
        ...
        END

SUB1    ...
        BSR     SUB2
        ...
        RTS

SUB2    ...
        RTS
```



LEGEND

A:	stack immediately before BSR SUB1
B:	stack immediately after BSR SUB1
C:	stack immediately after BSR SUB2
D:	stack immediately after first RTS
E:	stack immediately after second RTS



A simple example

- **Compute the average of an array of numbers**
  - In our case

$$\text{avg}(1,..,10) = \frac{1+..+10}{10} = 5.5$$

```
00001000                                1          ORG         $1000       ;data section
00001000  0000000A                      2 SIZE     DC.L        10          ;size of the array
00001004  00000001000000002000000       3 ARRAY    DC.L        1,2,3,4,5,6,7,8,9,10    ;array data
                                        4
00002000                                5          ORG         $2000       ;main program section
00002000  2038 1000                     6          MOVE.L      SIZE,D0     ;pass size in D0
00002004  41F8 1004                     7          LEA         ARRAY,A0    ;pass array address in A0
00002008  6100 0FF6                     8          BSR         AVG         ;call subroutine
0000200C  4E75                          9          RTS                     ;graceful exit
                                       10
00003000                               11          ORG         $3000       ;subroutine section
00003000  2200                         12 AVG      MOVE.L      D0,D1       ;D1 is the countdown index
00003002  4282                         13          CLR.L       D2          ;D2 returns the avg. value
00003004  D498                         14 REPT     ADD.L       (A0)+,D2    ;traverse the array
00003006  0441 0001                    15          SUBI        #1,D1       ;decrement counter
0000300A  66F8                         16          BNE         REPT        ;repeat while D1>0
0000300C  85C0                         17          DIVS        D0,D2       ;divide the sum by #elements
0000300E  4E75                         18          RTS                     ;return to the main program
00003010                               19          END
```

Run on sim68k

```
MC68000/ECB Simulator.
Copyright (C) Livadas and Ward, 1992.  Author Wayne Wolf
Version 2.3
SIM68000  2.3 > LO L6.S
SIM68000  2.3 > .PC 2000
SIM68000  2.3 > DF
PC=00002000 SR=0000=..0..... US=00007000 SS=00007E00
D0=F9315AE6 D1=390E2E5F D2=8E0FB027 D3=B78A9DAA
A0=718D7B96 A1=8C93757E A2=713179ED A3=6439C3E7
----------00002000 2038 1000              MOVE.L  $001000,D0

SIM68000  2.3 > T
PC=00002004 SR=8000=T.0..... US=00007000 SS=00007E00
D0=0000000A D1=390E2E5F D2=8E0FB027 D3=B78A9DAA
A0=718D7B96 A1=8C93757E A2=713179ED A3=6439C3E7
----------00002004 41F8 1004              LEA     $001004,A0

SIM68000  2.3 >
PC=00002008 SR=8000=T.0..... US=00007000 SS=00007E00
D0=0000000A D1=390E2E5F D2=8E0FB027 D3=B78A9DAA
A0=00001004 A1=8C93757E A2=713179ED A3=6439C3E7
----------00002008 6100 0FF6              BSR     $003000

SIM68000  2.3 >
PC=00003000 SR=8000=T.0..... US=00006FFC SS=00007E00
D0=0000000A D1=390E2E5F D2=8E0FB027 D3=B78A9DAA
A0=00001004 A1=8C93757E A2=713179ED A3=6439C3E7
----------00003000 2200                  MOVE.L  D0,D1

SIM68000  2.3 >
PC=00003002 SR=8000=T.0..... US=00006FFC SS=00007E00
D0=0000000A D1=0000000A D2=8E0FB027 D3=B78A9DAA
A0=00001004 A1=8C93757E A2=713179ED A3=6439C3E7
----------00003002 4282                  CLR.L   D2

SIM68000  2.3 >
PC=00003004 SR=8004=T.0..Z.. US=00006FFC SS=00007E00
D0=0000000A D1=0000000A D2=00000000 D3=B78A9DAA
A0=00001004 A1=8C93757E A2=713179ED A3=6439C3E7
----------00003004 D498                  ADD.L   (A0)+,D2
```

```
SIM68000  2.3 >
PC=00003006 SR=8000=T.0..... US=00006FFC SS=00007E00
D0=0000000A D1=0000000A D2=00000001 D3=B78A9DAA
A0=00001008 A1=8C93757E A2=713179ED A3=6439C3E7
----------00003006 0441 0001              SUB.W   #$0001,D1

SIM68000  2.3 >
PC=0000300A SR=8000=T.0..... US=00006FFC SS=00007E00
D0=0000000A D1=00000009 D2=00000001 D3=B78A9DAA
A0=00001008 A1=8C93757E A2=713179ED A3=6439C3E7
----------0000300A 66F8                  BNE     $003004

SIM68000  2.3 >
PC=00003004 SR=8000=T.0..... US=00006FFC SS=00007E00
D0=0000000A D1=00000009 D2=00000001 D3=B78A9DAA
A0=00001008 A1=8C93757E A2=713179ED A3=6439C3E7
----------00003004 D498                  ADD.L   (A0)+,D2

SIM68000  2.3 >
PC=00003006 SR=8000=T.0..... US=00006FFC SS=00007E00
D0=0000000A D1=00000009 D2=00000003 D3=B78A9DAA
A0=0000100C A1=8C93757E A2=713179ED A3=6439C3E7
----------00003006 0441 0001              SUB.W   #$0001,D1

SIM68000  2.3 >
PC=0000300A SR=8000=T.0..... US=00006FFC SS=00007E00
D0=0000000A D1=00000008 D2=00000003 D3=B78A9DAA
A0=0000100C A1=8C93757E A2=713179ED A3=6439C3E7
----------0000300A 66F8                  BNE     $003004

SIM68000  2.3 >
PC=00003004 SR=8000=T.0..... US=00006FFC SS=00007E00
D0=0000000A D1=00000008 D2=00000003 D3=B78A9DAA
A0=0000100C A1=8C93757E A2=713179ED A3=6439C3E7
----------00003004 D498                  ADD.L   (A0)+,D2

SIM68000  2.3 >
PC=00003006 SR=8000=T.0..... US=00006FFC SS=00007E00
D0=0000000A D1=00000008 D2=00000006 D3=B78A9DAA
A0=00001010 A1=8C93757E A2=713179ED A3=6439C3E7
----------00003006 0441 0001              SUB.W   #$0001,D1
```

```
SIM68000  2.3 >
PC=0000300A SR=8000=T.0..... US=00006FFC SS=00007E00
D0=0000000A D1=00000007 D2=00000006 D3=B78A9DAA
A0=00001010 A1=8C93757E A2=713179ED A3=6439C3E7
----------0000300A 66F8                  BNE     $003004

...

SIM68000  2.3 >
PC=00003004 SR=8000=T.0..... US=00006FFC SS=00007E00
D0=0000000A D1=00000001 D2=0000002D D3=B78A9DAA
A0=00001028 A1=8C93757E A2=713179ED A3=6439C3E7
----------00003004 D498                  ADD.L   (A0)+,D2

SIM68000  2.3 >
PC=00003006 SR=8000=T.0..... US=00006FFC SS=00007E00
D0=0000000A D1=00000001 D2=00000037 D3=B78A9DAA
A0=0000102C A1=8C93757E A2=713179ED A3=6439C3E7
----------00003006 0441 0001              SUB.W   #$0001,D1

SIM68000  2.3 >
PC=0000300A SR=8004=T.0..Z.. US=00006FFC SS=00007E00
D0=0000000A D1=00000000 D2=00000037 D3=B78A9DAA
A0=0000102C A1=8C93757E A2=713179ED A3=6439C3E7
----------0000300A 66F8                  BNE     $003004

SIM68000  2.3 >
PC=0000300C SR=8004=T.0..Z.. US=00006FFC SS=00007E00
D0=0000000A D1=00000000 D2=00000037 D3=B78A9DAA
A0=0000102C A1=8C93757E A2=713179ED A3=6439C3E7
----------0000300C 85C0                  DIVS    D0,D2

SIM68000  2.3 >
PC=0000300E SR=8000=T.0..... US=00006FFC SS=00007E00
D0=0000000A D1=00000000 D2=00050005 D3=B78A9DAA
A0=0000102C A1=8C93757E A2=713179ED A3=6439C3E7
----------0000300E 4E75                  RTS

SIM68000  2.3 >
PC=0000200C SR=8000=T.0..... US=00007000 SS=00007E00
D0=0000000A D1=00000000 D2=00050005 D3=B78A9DAA
A0=0000102C A1=8C93757E A2=713179ED A3=6439C3E7
----------0000200C 4E75                  RTS
```

AVG(1,..,10) = 5.5

A look at the STACK on sim68k

```
MC68000/ECB Simulator.
Copyright (C) Livadas and Ward, 1992.  Author Wayne Wolf
Version 2.3
SIM68000  2.3 > LO L6.S
SIM68000  2.3 > BR 2008

BREAKPOINTS
002008     002008

SIM68000  2.3 > BR 300E

BREAKPOINTS
002008     002008
00300E     00300E

SIM68000  2.3 > GO 2000

AT BREAKPOINT
PC=00002008 SR=8000=T.0.... US=00007000 SS=00007E00
D0=0000000A D1=AEFBC1E5 D2=5612A2AA D3=862C6564
A0=00001004 A1=47466DA7 A2=340641B8 A3=299A886F
----------00002008 6100 0FF6              BSR     $003000


SIM68000  2.3 > MD 6FF0 20

006FF0   00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ................
007000   00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ................

SIM68000  2.3 > T

PC=00003000 SR=8000=T.0.... US=00006FFC SS=00007E00
D0=0000000A D1=AEFBC1E5 D2=5612A2AA D3=862C6564
A0=00001004 A1=47466DA7 A2=340641B8 A3=299A886F
----------00003000 2200                MOVE.L  D0,D1

SIM68000  2.3 > MD 6FF0 20

006FF0   00 00 00 00 00 00 00 00   00 00 00 00 00 20 0C   ..............  .
007000   00 00 00 00 00 00 00 00   00 00 00 00 00 00 00   ................
```

```
SIM68000  2.3 > GO

AT BREAKPOINT
PC=0000300E SR=8000=T.0.... US=00006FFC SS=00007E00
D0=0000000A D1=00000000 D2=00050005 D3=DFB2B4F7
D4=17878BD5 D5=47886B47 D6=2B9E229F D7=402E85E6
A0=0000102C A1=C2B6AEC4 A2=B78BD3A6 A3=7FDC3CCC
A4=9E3FBED4 A5=B244AB91 A6=AD715F12 A7=00006FFC
----------0000300E 4E75              RTS


SIM68000  2.3 > MD 6FF0 20

006FF0   00 00 00 00 00 00 00 00   00 00 00 00 00 20 0C   ..............  .
007000   00 00 00 00 00 00 00 00   00 00 00 00 00 00 00   ................

SIM68000  2.3 > T

PC=0000200C SR=8000=T.0.... US=00007000 SS=00007E00
D0=0000000A D1=00000000 D2=00050005 D3=DFB2B4F7
D4=17878BD5 D5=47886B47 D6=2B9E229F D7=402E85E6
A0=0000102C A1=C2B6AEC4 A2=B78BD3A6 A3=7FDC3CCC
A4=9E3FBED4 A5=B244AB91 A6=AD715F12 A7=00007000
----------0000200C 4E75              RTS

SIM68000  2.3 > MD 6FF0 20

006FF0   00 00 00 00 00 00 00 00   00 00 00 00 00 20 0C   ..............  .
007000   00 00 00 00 00 00 00 00   00 00 00 00 00 00 00   ................

SIM68000  2.3 > T

PC=0000200C SR=8000=T.0..... US=00007000 SS=00007E00
D0=0000000A D1=00000000 D2=00050005 D3=DFB2B4F7
D4=17878BD5 D5=47886B47 D6=2B9E229F D7=402E85E6
A0=0000102C A1=C2B6AEC4 A2=B78BD3A6 A3=7FDC3CCC
A4=9E3FBED4 A5=B244AB91 A6=AD715F12 A7=00007000
----------0000200C 4E75                RTS
```

Parameter passing

- **Two ways to pass parameters to a subroutine**
  - **By value**: the actual data is passed in a register
  - **By reference**: the address in memory of the parameter is passed in a register
- **In the previous example (`BSR AVG`)**
  - The number of items in the array (**`SIZE`**) was passed by value since it was loaded on D0 (**`MOVE.L SIZE,D0`**)
  - The items (**`ARRAY`**) were passed by reference since their starting address was loaded on A0 (**`LEA ARRAY,A0`**)
- **These parameter passing methods are limiting**
  - They do not allow re-entrant code
  - Using the stack is the preferred method (next lecture)