EE4390 Microprocessors

Lessons 11, 12

Advanced Assembly Programming

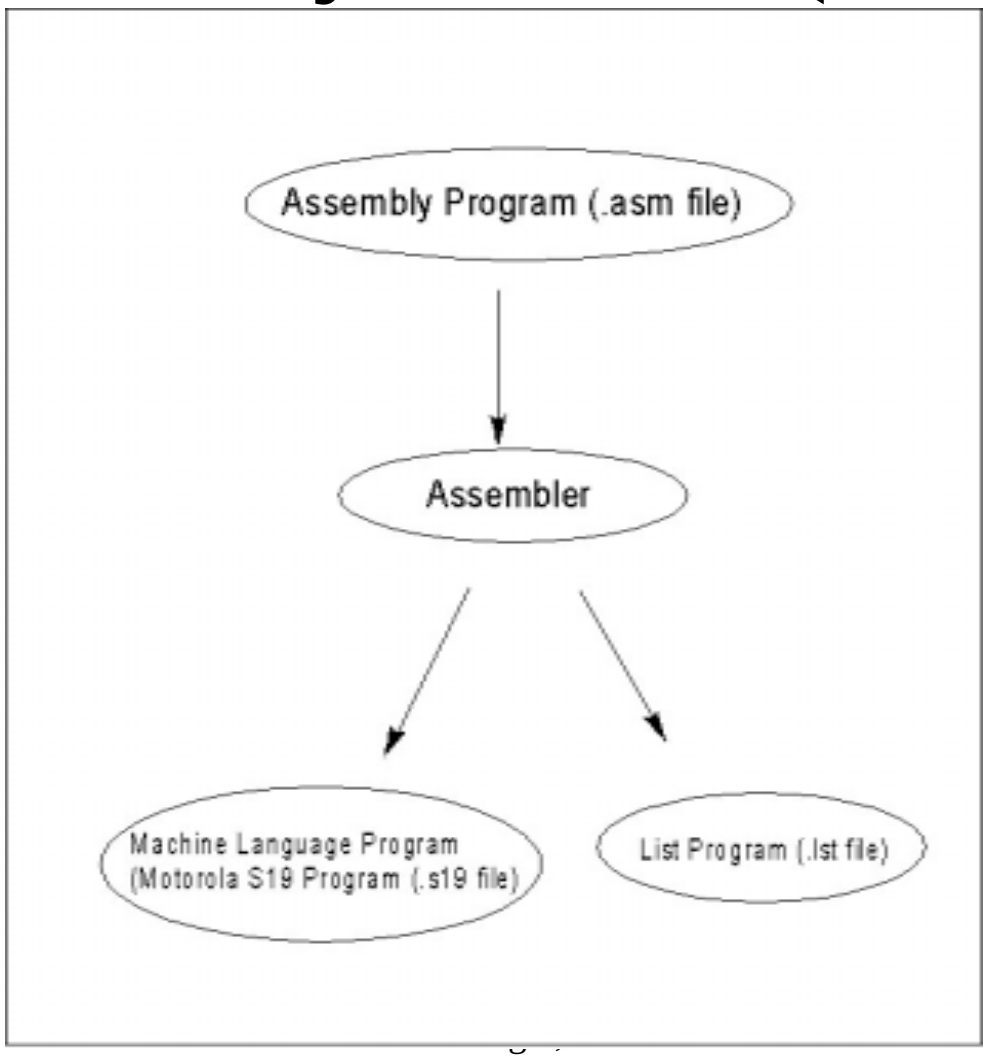
Overview

- Assembly Process
- Loops
- Stack
- Subroutines
- 68HC12 D-BUG12 Utility Subroutines
- Programming Modules - Unified Modeling Language
- Programming Techniques

Assembly Process

- Assembler converts assembly language source file (*.asm) to machine language (1's and 0's)
- Motorola machine language file is an *.S19
- As part of the assembly process one can generate a list file (*.lst)
  - contains address for each instruction
  - provides contents of each address

Assembly Process (cont)



Loops

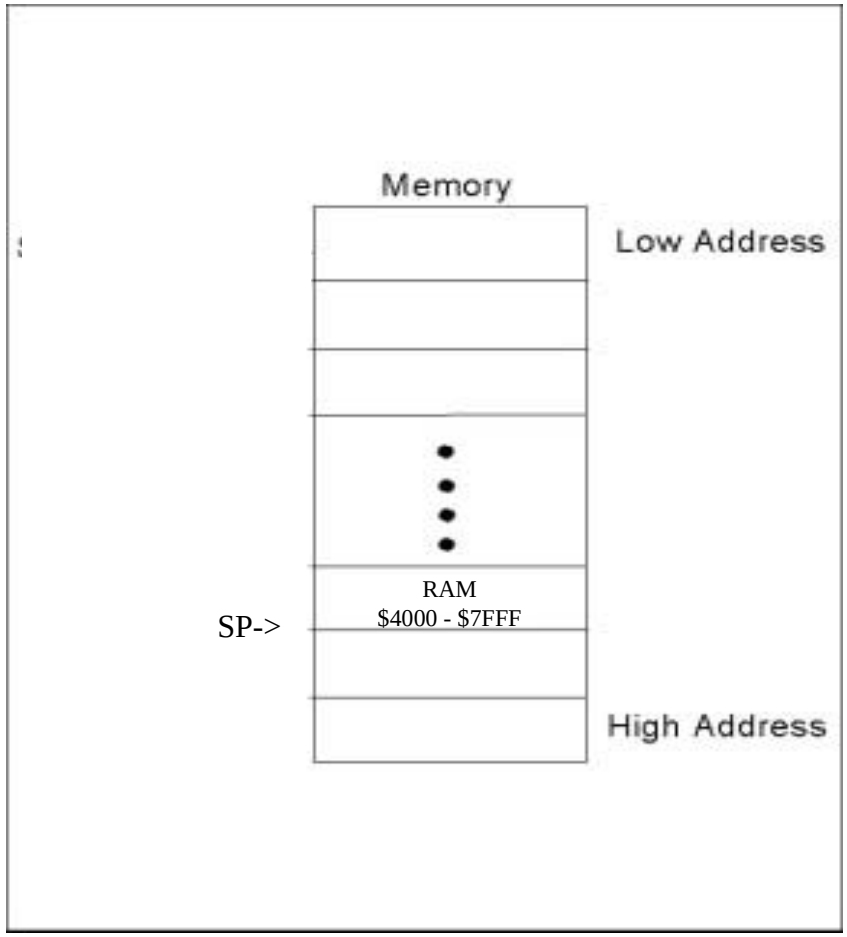
- Software construct to perform code iterations
  - HOL: for(i=0;i<=10;i++){do something;}
- Loop construct - 5 step process
  - initialize loop counter
  - perform specified operation, increment (or decrement) loop counter
  - compare counter with limit
  - exit if the value of the loop counter is greater (less) than the upper (lower) limit

Loops (cont)

Label	**Op-Code**	**Operand(s)**	**Comment**
	LDX	#$1000	;initialize index X
	LDAB	SUM	;load sum to B
	LDAA	#$00	;initialize loop ctr
CHECK	CMPA	#$0A	;10 scores yet?
	BEQ	STOP	;if so, exit loop
	ADAB	0,X	;add to sum
	INX		;inc index X
	INCA		;inc loop counter
	BRA	CHECK	;start loop again
STOP	STAB	SUM	;save the result

Stack

- Portion of RAM set aside for temporary data storage
- “Top of Stack” refers to address of last element loaded on stack
- Stack Pointer (SP) keeps track of stack top



Stack (cont)

- Programmer dictates initial contents of stack
  - Declare as last location in largest RAM space plus 1
  - For the “A4”, this is $8000 (user RAM: $4000 - $7FFF)
- First-in-last-out data structure
- Temporary storage during program execution
  - PUSH: place register contents on stack
  - PULL: place stack location contents in register

  EX] PSHA, PSHX, PULB, PULY

- Need to PSH and PUL in opposite order to retain original register values

Stack (cont)

- Stack Initialization

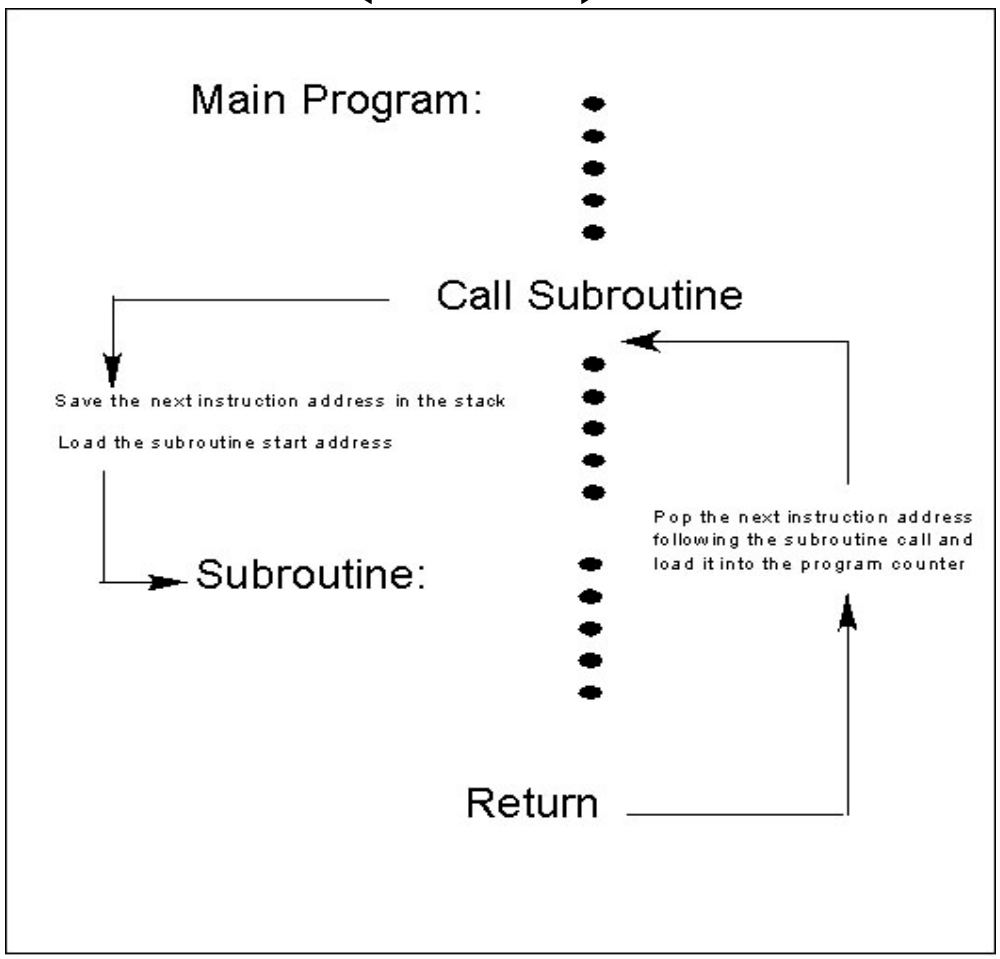
```
;directives
STACKTOP = $8000

;code
        LDS  #STACKTOP
```

Subroutines

- Separate, independent module of program, performs a specific task
  - shortens code, provide reusable “tools”
- Good subroutine:
  - independence: does not depend on other code
  - registers: stores/restores key registers upon entrance/exit using PSH and PUL commands
  - data and code independent: local variables, do not use direct and extended addressing modes

Subroutines (cont)

- BSR vs JSR
- RTS
- Return address automatically PSH and PUL to stack
- Key registers must be PSH and PUL by **programmer**



Subroutines (cont)

```
STACKTOP = $8000
_main::
        LDS     #STACKTOP       ;initialize SP
        JSR     DOIT            ;jump to subroutine
HERE    BRA     HERE

DOIT:   PSHA                    ;store key registers on stack
        PSHB
        :                       ;subroutine actions
        PULB                    ;restore key register values
        PULA
        RTS                     ;return from subroutine
```

D-BUG12 Utility Routines

TABLE 3.1 D-BUG12 UTILITY ROUTINES TABLE.

Routines	Vector Address	Function
main()	$FE00	Starts the D-Bug12 monitor porgram
getchar()	$FE02	Fetches a character from the keyboard
putchar()	$FE04	Displays a character on the screen
printf()	$FE06	Displays a formatted string on the screen
GetCmdLine()	$FE08	Fetches a command from the keyboard
sscanhex()	$FE0A	Converts a hexadecimal number to an integer
isxdigit()	$FE0C	Checks for a hexadecimal digit
toupper()	$FE0E	Converts a lowercase character to the uppercase character
isalpha()	$FE10	Checks for an alphabet character
strlen()	$FE12	Returns the length of a string
strcpy()	$FE14	Copies a string to another string
out2hex()	$FE16	Outputs a byte of a two hexadecimal number
out4hex()	$FE18	Outputs two bytes of a four hexadecimal number
SetUserVector()	$FE1A	Setup user-specified interrupt service routine address
WriteEEByte()	$FE1C	Write a byte to on-chip EEPROM
EraseEE()	$FE1E	Erase a block of on-chip EEPROM
ReadMem()	$FE20	Read data from memory
WriteMem()	$FE22	Write data to memory

[2]An interrupt service routine is a type of subroutine executed when a special signal, called an *interrupt*, is detected by a controller. The special signal can be initiated by software or hardware. An example of a hardware interrupt is a signal generated by an external device connected to the controller requesting a service of the controller.

D-BUG12 Utility Routines (cont)

- For D-BUG12 routines, the 68HC12 insists on using stack to transfer parameter variables
  - treat parameters as 16-bit values
- Parameters must be pushed onto the stack starting with parameter n to 1
- Load vector address of the subroutine in index register X

D-BUG12 Utility Routines (cont)

EX] Use out2hex() utility subroutine to display $45 on the computer screen

– store $0045 to stack before calling subroutine

```
LDD   #$45          ;value for display
PSHD
LDX   #$FE16        ;16-bit addr of subr
JSR   0,X           ;call the subroutine
PULX                ;clean up the stack
```