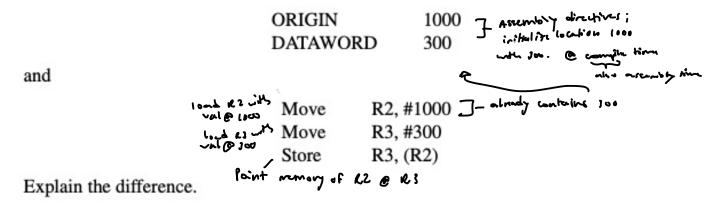
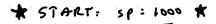
- 2.4 [E] Registers R4 and R5 contain the decimal numbers 2000 and 3000 before each of the following addressing modes is used to access a memory operand. What is the effective address (EA) in each case? 2.5 2.5000
  - (a) 12(R4) EA = R4 +12 = 2000+12 = 2012
  - (b) (R4,R5) (A: R4+R5 = 2000+8000 = 5000
  - (c) 28(R4,R5) EA = RYIRS + 28 : 5028
  - (d) (R4)+ 6A: R4+4: 2004
  - (e) -(R4) 64 : K4 4 : 1996
- 2.12 [E] Both of the following statement segments cause the value 300 to be stored in location 1000, but at different times.



## Difference:

- · first segment was directives to initiallise memory at assembly-time
- second segment dynamically performs the storage at run-time
- 2.20 [M] Show the processor stack contents and the contents of the stack pointer, SP, immediately after each of the following instructions in the program in Figure 2.18 is executed. Assume that [SP] = 1000 at Level 1, before execution of the calling program begins.
  - (a) The second Store instruction in the subroutine Sp: 972; shule: [..., RS, R4, R3, R2, 100, N, Now 1]
  - (b) The last Load instruction in the subroutine 58: 972; sheek: [..., as a4, a3, a2, 100, 5000, NUM!]
  - (c) The last Store instruction in the calling program 38: 992; shows C... som, work



2.26 [M] The dot-product computation is discussed in Section 2.12.1. This type of computation can be used in the following signal-processing task. An input signal time sequence IN(0), IN(1), IN(2), IN(3), ..., is processed by a 3-element weight vector (WT(0), WT(1),

WT(2)) = (1/8, 1/4, 1/2) to produce an output signal time sequence OUT(0), OUT(1), OUT(2), OUT(3), . . . , as follows:

$$\begin{aligned} & \text{OUT}(0) = \text{WT}(0) \times \text{IN}(0) + \text{WT}(1) \times \text{IN}(1) + \text{WT}(2) \times \text{IN}(2) \\ & \text{OUT}(1) = \text{WT}(0) \times \text{IN}(1) + \text{WT}(1) \times \text{IN}(2) + \text{WT}(2) \times \text{IN}(3) \\ & \text{OUT}(2) = \text{WT}(0) \times \text{IN}(2) + \text{WT}(1) \times \text{IN}(3) + \text{WT}(2) \times \text{IN}(4) \\ & \text{OUT}(3) = \text{WT}(0) \times \text{IN}(3) + \text{WT}(1) \times \text{IN}(4) + \text{WT}(2) \times \text{IN}(5) \\ & \vdots \end{aligned}$$

All signal and weight values are 32-bit signed numbers. The weights, inputs, and outputs, are stored in the memory starting at locations WT, IN, and OUT, respectively. Write a RISC-style program to calculate and store the output values for the first *n* outputs, where *n* is stored at location N.

Hint: Arithmetic right shifts can be used to do the multiplications.

```
Assembly:
LOAD RI, N
land RZ, IN
load 12, WT
long Ry, OUT
move LS, #0
    lood RG, LR2)
    1, ed P7, 4(RZ)
    1006 RS, 8(A2)
    move 11. ±0
    MAR A40, $0
     more ell, 40
     non til, flb
    1 hift Right Anthoreti R12, #3
     mon RISIAT
     shift Right Anthrobic R13, #2
     MON PIN, ES
     Mist Right Anthoretic Rem ,#1
     Add RIS, RIZ, AIS
     Add the lost the
     Stor LLS, (M)
     Add R4 R4 #4
     Add Rs, Rs, #1
```