

1. 0. MOV R0, R0, #0xFFFFFFFF  
1. EOR R0, R0, #0xFFFFFFFF

2. 1. can have separate RAM and ROM memory devices

3. 2. ADDLT R0, R0, #1

4. 3. no bytes

5. 0. an assembler error

6. 2. the processor had a 40-bit address bus

7. 1. represents an address

8. STR R2, [R1], #4

9. The end of the code in your file does not have any special significance to the processor. It is simply a very sophisticated FSM that keeps running. Execution would continue with a fetch from the next location in memory. The processor would continue to run unless it encountered an undefined instruction, or caused an abort condition.

10. Strengths:

Simple serial interface using just a few pins, already included on chip, only small connector required on board, full control over CPU pins and registers

Weaknesses:

Serial connection is relatively slow, must clock in/out entire scan chain, must halt processor clock when reading/writing scan chain

11. They don't differ at all! The ARM7 uses memory-mapped I/O, so I/O is accessed using the same load/store instructions as the rest of memory.

12. SUB R0, PC, #12

13. R1 = 0x0000000F

R2 = 0x08090A0B

R3 = 0x04050607

R4 = 0x12345668

14. FIQ mode. When entering FIQ mode, you do not have to save R8-R12 since they are not used by any other mode. This is part of what makes FIQ the *fast* interrupt mode.

15.    MOV<sub>S</sub>        R0, R0               ; test value  
        RSBMI      R0, #0             ; if negative, negate to get absolute value
16.    MOV         R0, #0x40000003   ; load address since it is a valid immediate  
        LDRB      R0, [R0]           ; get byte from memory
17.    MOV         R0, R0, LSL #24   ; move bit 7 to MSb  
        MOV         R0, R0, ASR #24   ; move back to bit 7 with sign extension
18.    TST         R0, #3             ; test bits 1:0  
        ORREQ     R0, R0, #3         ; could also EOR with 3 or ADD 3
19.    AND         R1, R0, #0xFF      ; mask off LSB into R1  
        CMP         R1, #0x35         ; is it the value  
        EOREQ     R0, R0, #0xFF000000 ; if so, toggle MSB
20.    0x00000-0x1FFFF → 0x20000 locations,  $0x20000 = 2^{17}$  → need 17 address lines  
 $2^{17} = 2^7 \cdot 2^{10} = 128 \text{ KB}$   
 0x10000-0x1FFFF → 0x10000 locations,  $0x10000B = 2^{16}B = 64\text{KB}$   
 32KB →  $2^5 \cdot 2^{10}B = 2^{15}B = 0x8000$  locations, so last address is 0x07FFF