CSE 331L: Microprocessor Interfacing and Embedded Systems Lab

Summer 2025

Arm Assembly (Part-2)

Class # 03

Recap

- Instructions (Types and Parts)
- MOV
- ADD
- SUB
- MUL
- Memory Segments(8086 vs Arm)
- Variables
- LDR
- STR

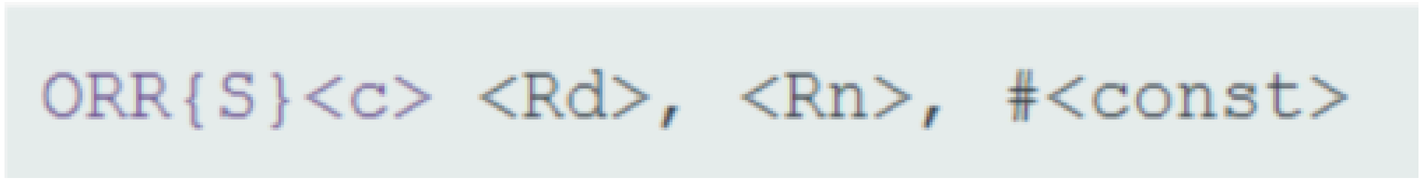
Logical Operators

- ORR (Logical OR)
- AND (Logical AND)
- EOR (Exclusive- OR)
- MVN (Logical NOT / Negation)

Reference

ORR immediate

Bitwise OR (immediate) performs a bitwise (inclusive) OR of a register value and an immediate value, and writes the result to the destination register. It can optionally update the condition flags based on the result.

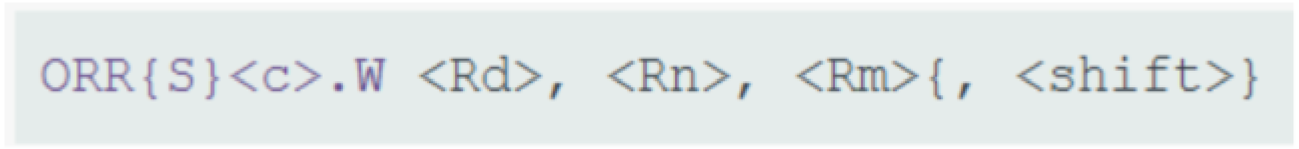


```
ORR{S}<c> <Rd>, <Rn>, #<const>
```

Reference

ORR

Bitwise OR (register) performs a bitwise (inclusive) OR of a register value and an optionally-shifted register value, and writes the result to the destination register. It can optionally update the condition flags based on the result.

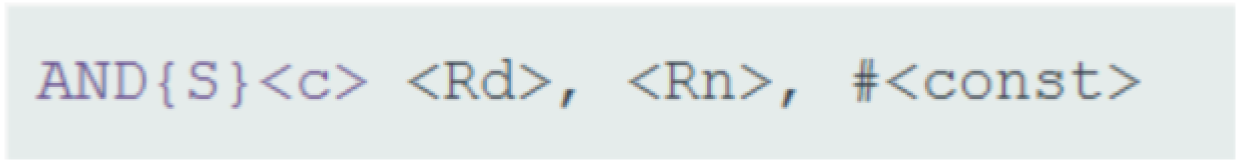


```
ORR{S}<c>.W <Rd>, <Rn>, <Rm>{, <shift>}
```

Reference

AND immediate

This instruction performs a bitwise AND of a register value and an immediate value, and writes the result to the destination register.

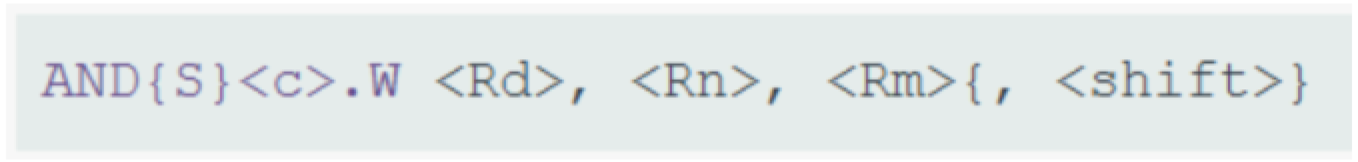


```
AND{S}<c> <Rd>, <Rn>, #<const>
```

Reference

AND

This instruction performs a bitwise AND of a register value and an optionally-shifted register value, and writes the result to the destination register. It can optionally update the condition flags based on the result.

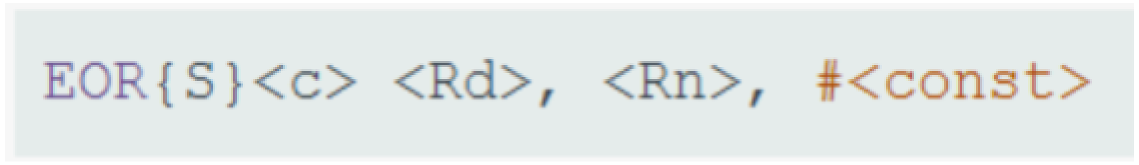


```
AND{S}<c>.W <Rd>, <Rn>, <Rm>{, <shift>}
```

Reference

EOR immediate

Bitwise Exclusive OR (immediate) performs a bitwise Exclusive OR of a register value and an immediate value, and writes the result to the destination register. It can optionally update the condition flags based on the result.

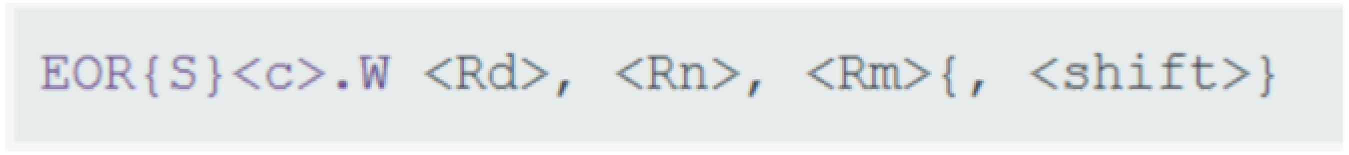


```
EOR{S}<c> <Rd>, <Rn>, #<const>
```

Reference

EOR

Bitwise Exclusive OR (register) performs a bitwise Exclusive OR of a register value and an optionally-shifted register value, and writes the result to the destination register. It can optionally update the condition flags based on the result.

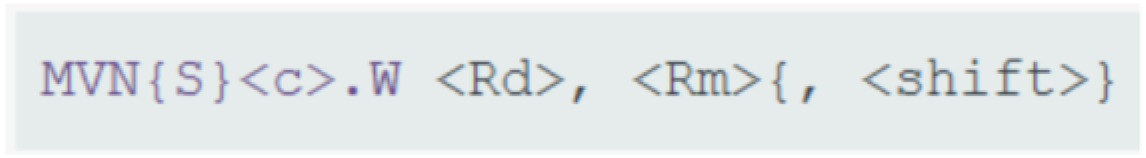


```
EOR{S}<c>.W <Rd>, <Rn>, <Rm>{, <shift>}
```

Reference

MVN

Bitwise NOT (register) writes the bitwise inverse of a register value to the destination register. It can optionally update the condition flags based on the result.

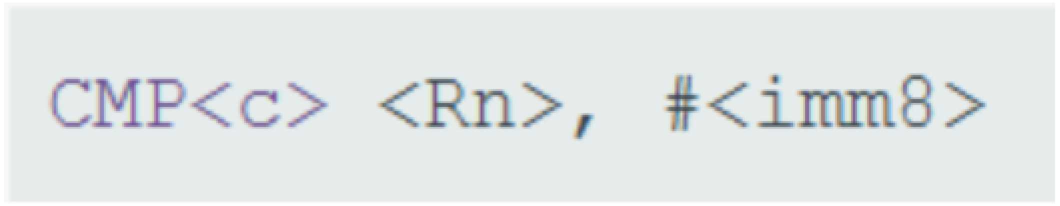


```
MVN{S}<c>.W <Rd>, <Rm>{, <shift>}
```

Reference

CMP immediate

Compare (immediate) subtracts an immediate value from a register value. It updates the condition flags based on the result, and discards the result.

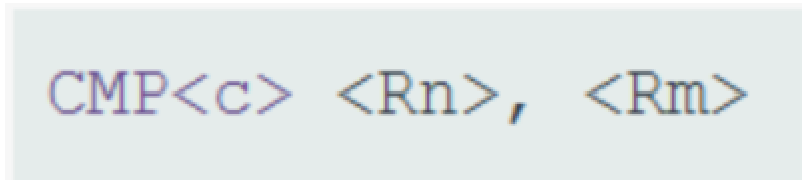


```
CMP<c> <Rn>, #<imm8>
```

Reference

CMP

Compare (register) subtracts an optionally-shifted register value from a register value. It updates the condition flags based on the result, and discards the result.

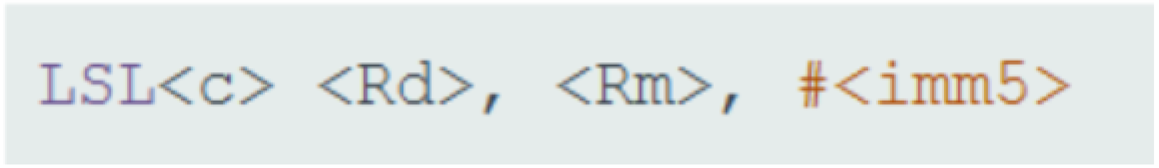


```
CMP<c> <Rn>, <Rm>
```

Reference

LSL immediate

Logical Shift Left (immediate) shifts a register value left by an immediate number of bits, shifting in zeros, and writes the result to the destination register. It can optionally update the condition flags based on the result.

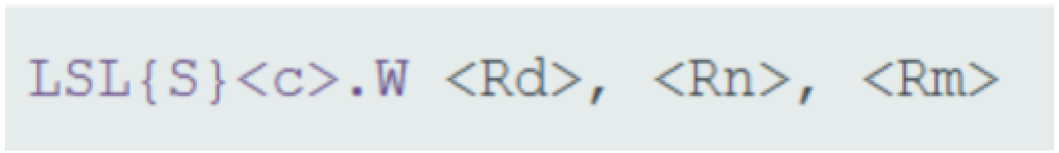


```
LSL<c> <Rd>, <Rm>, #<imm5>
```

Reference

LSL

Logical Shift Left (register) shifts a register value left by a variable number of bits, shifting in zeros, and writes the result to the destination register. The variable number of bits is read from the bottom byte of a register. It can optionally update the condition flags based on the result.

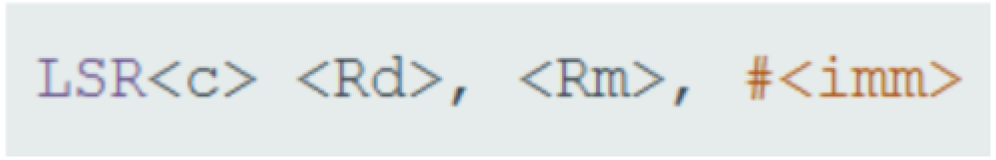


```
LSL{S}<c>.W <Rd>, <Rn>, <Rm>
```

Reference

LSR immediate

Logical Shift Right (immediate) shifts a register value right by an immediate number of bits, shifting in zeros, and writes the result to the destination register. It can optionally update the condition flags based on the result.

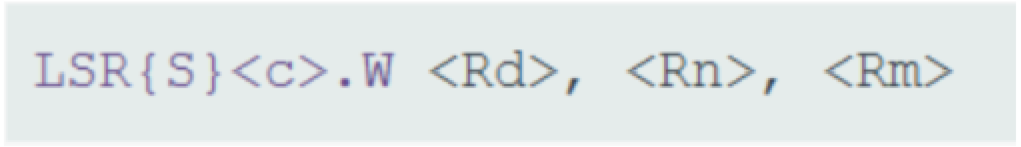


```
LSR<c> <Rd>, <Rm>, #<imm>
```

Reference

LSR

Logical Shift Right (register) shifts a register value right by a variable number of bits, shifting in zeros, and writes the result to the destination register. The variable number of bits is read from the bottom byte of a register. It can optionally update the condition flags based on the result.



```
LSR{S}<c>.W <Rd>, <Rn>, <Rm>
```

Reference

ROR (Rotate Right) immediate

Rotate Right (immediate) provides the value of the contents of a register rotated by a constant value. The bits that are rotated off the right end are inserted into the vacated bit positions on the left. It can optionally update the condition flags based on the result.

```
ROR{S}<c> <Rd>, <Rm>, #<imm>
```

Reference

ROR

Any change?

```
ROR{S}<c>.W <Rd>, <Rn>, <Rm>
```

Reference

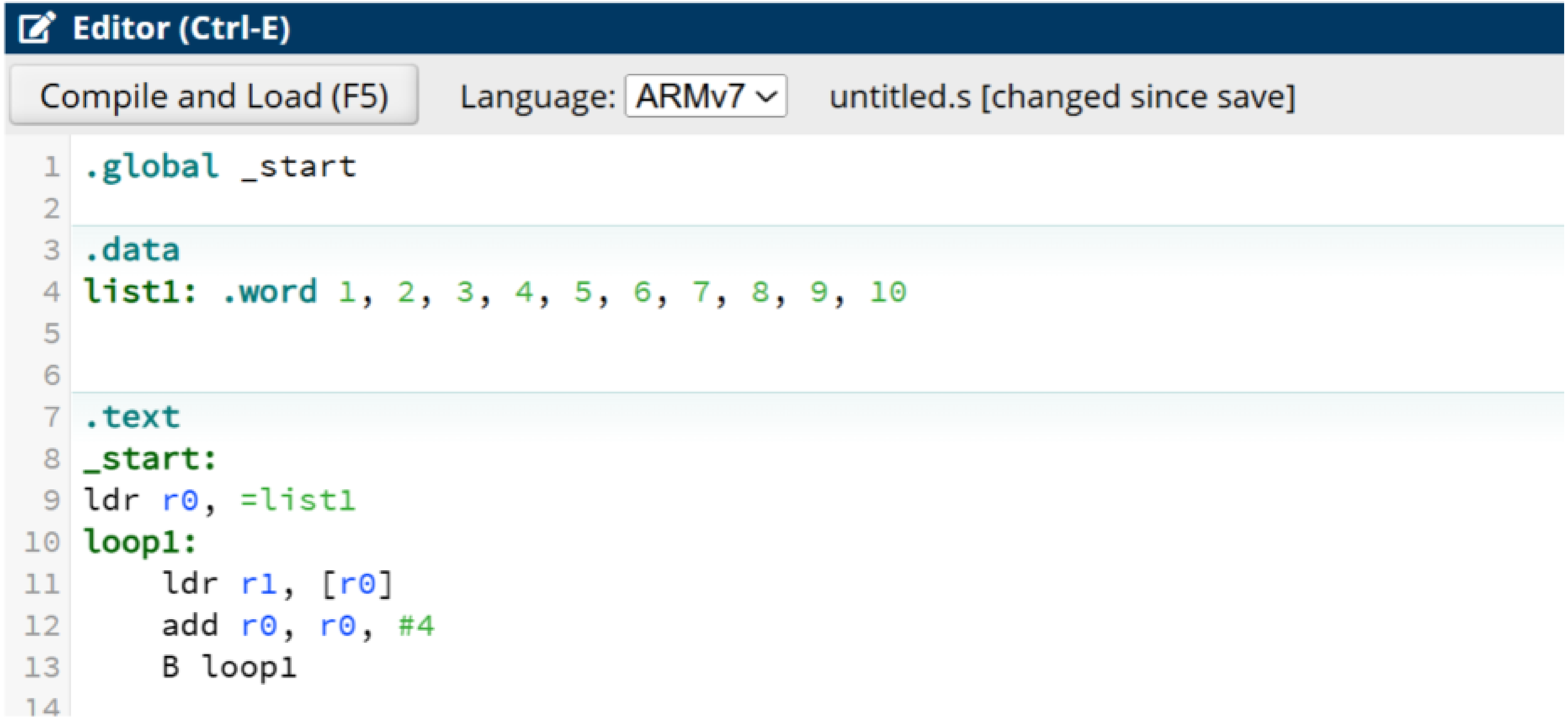
Question

- Difference between ROL and ROR?

Arrays in assembly

- Where do we store array?
- How to declare?

Declaring an Array in assembly



Editor (Ctrl-E)

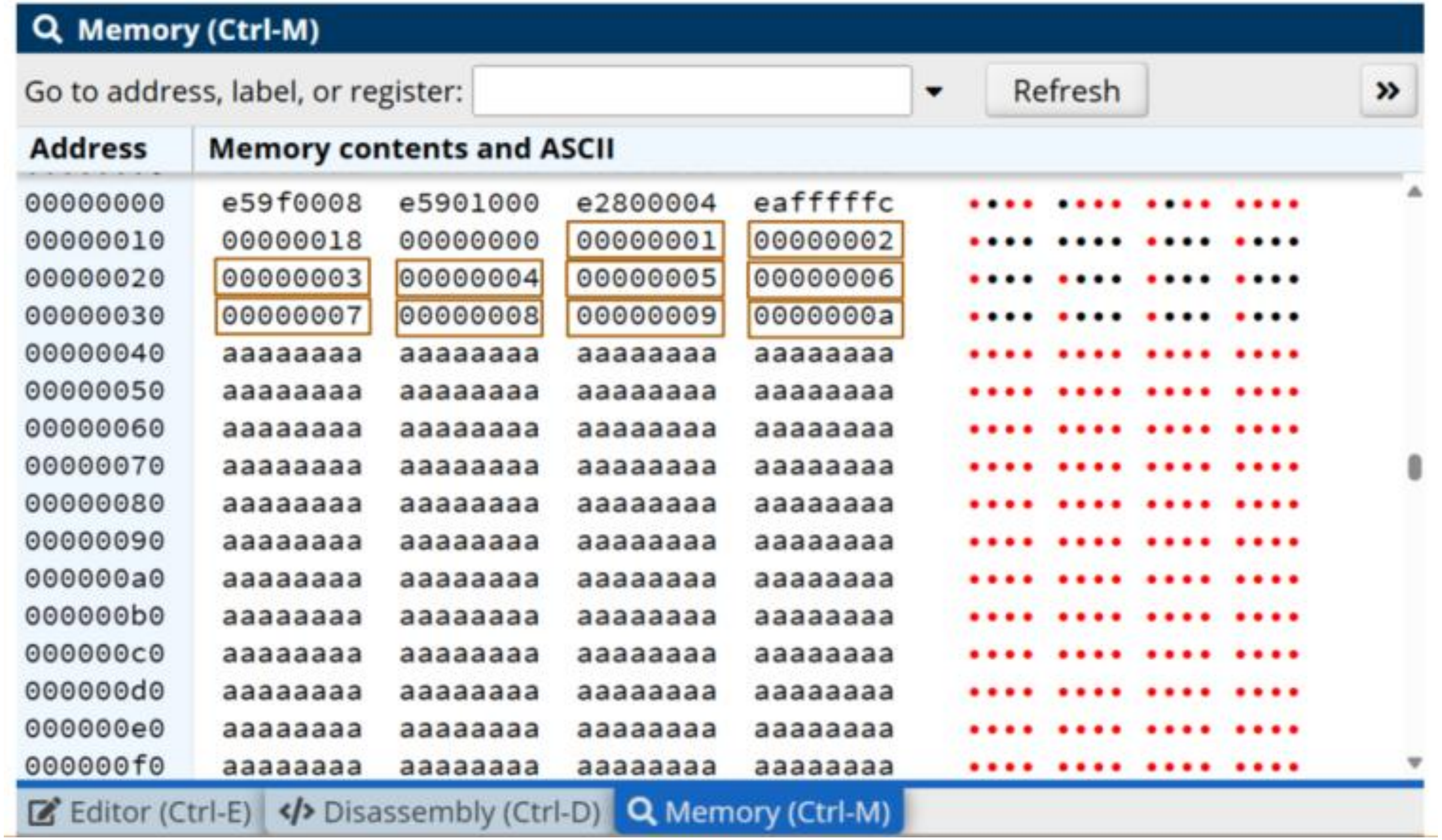
Compile and Load (F5) Language: ARMv7 untitled.s [changed since save]

```
.global _start

.data
list1: .word 1, 2, 3, 4, 5, 6, 7, 8, 9, 10


.text
_start:
ldr r0, =list1
loop1:
    ldr r1, [r0]
    add r0, r0, #4
    B loop1
```

Declaring an Array in assembly



Memory (Ctrl-M)

Go to address, label, or register: Refresh

```
Address    Memory contents and ASCII
00000000   e59f0008  e5901000  e2800004  eaffff fc
00000010   00000018  00000000  00000001  00000002
00000020   00000003  00000004  00000005  00000006
00000030   00000007  00000008  00000009  0000000a
00000040   aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
00000050   aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
00000060   aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
00000070   aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
00000080   aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
00000090   aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
000000a0   aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
000000b0   aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
000000c0   aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
000000d0   aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
000000e0   aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
000000f0   aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
```