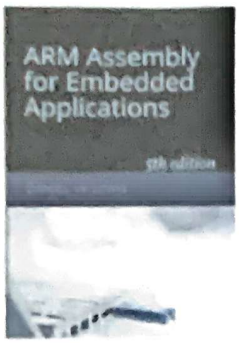


Programming Lab 4C

Optimization

Topics: Address alignment, address and data dependencies, overlapped execution.

Click to download Lab4C-Main.c

Prerequisite Reading: Chapters 1-4
Revised: February 17, 2021

Run-time performance can be adversely affected due to improper address alignment, the sequence in which instructions are executed, or by not overlapping the execution of floating-point divide or square root instructions with the execution of integer instructions. Faster execution can sometimes be achieved by a simple rearrangement of instructions. In this lab you will create the following functions and measure their execution time from which you can determine the corresponding performance penalties in clock cycles per instruction[1].

Address Alignment: Extra memory cycles are required when 16-bit operands are not located at addresses that are a multiple of 2, and 32-bit or 64-bit operands are not at a multiple of 4.

```
FullWordAccess:
    .rept       100
    LDR         R1,[R0]
    .endr
    BX          LR
```

```
HalfWordAccess:
    .rept       100
    LDRH        R1,[R0]
    .endr
    BX          LR
```

Address Dependency: Any load or store instruction whose address depends on a register that was modified by the preceding instruction will always be delayed while the register is updated.

```
AddressDependency:
    .rept       100
    LDR         R1,[R0]
    LDR         R0,[R1]
    .endr
    BX          LR
```

```
NoAddressDependency:
    .rept       100
    LDR         R1,[R0]
    LDR         R2,[R0]
    .endr
    BX          LR
```

Data Dependency: A floating-point instruction (e.g., `VADD.F32`) will always be delayed if one of its input operands is the output of the preceding floating-point arithmetic instruction. *(See the footnote[2] about the VMOV instructions.)*

```
DataDependency:
    .rept       100
    VADD.F32    S1,S0,S0
    VADD.F32    S0,S1,S1
    .endr
    VMOV        S1,S0
    BX          LR
```

```
NoDataDependency:
    .rept       100
    VADD.F32    S1,S0,S0
    VADD.F32    S2,S0,S0
    .endr
    VMOV        S1,S0
    BX          LR
```

Concurrent Execution: The slow execution of `VDIV.F32` or `VSQRT.F32` may be overlapped with a sequence of several integer-only instructions. Determine amount of overlap possible by increasing the repetitions of the `NOP` until the displayed execution time of the function begins to increase. *(See footnote[2] about the VMOV instruction.)*

```
VDIVOverlap:
    VDIV.F32    S2,S1,S0
    .rept       1
    NOP
    .endr
    VMOV        S3,S2
    BX          LR
```

Analyze the measured execution times and source code to determine:

Half word address = `X...XX1` penalty: __1__ cycles/instruction

Full word address = `X...X01` penalty: __2__ cycles/instruction

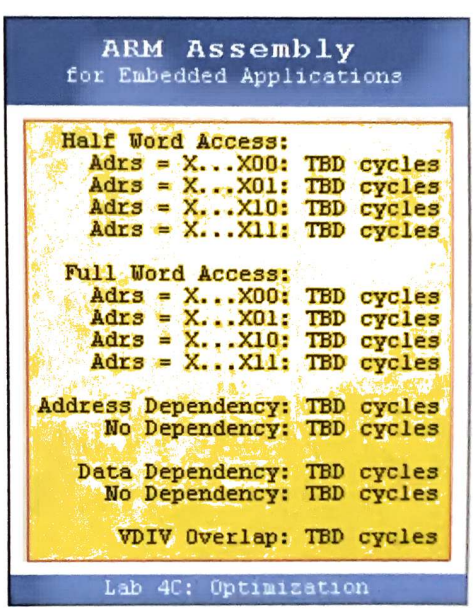
Full word address = `X...X10` penalty: __1__ cycles/instruction

Full word address = `X...X11` penalty: __2__ cycles/instruction

Address dependency penalty: __1__ cycles/instruction

Data dependency penalty: __1__ cycles/instruction

Maximum `VDIV/VSQRT` overlap: __14__ clock cycles



---

[1] *The "TBD's" shown in the figure will be replaced by the cycle counts required to execute the assembly language functions.*

[2] *The `VMOV` instruction in functions `DataDependency` and `VDIVOverLap` are used to force the previous floating-point instruction to complete before executing the `BX LR` return. The `VMOV` in `NoDataDependency` provides measurement consistency.*