|  |  |  |  |
| --- | --- | --- | --- |
| **Subject** | **Subject Code** | **Section** | **Google Class Room Link** |
| ARM Processor | EC4C02 | A | https://meet.google.com/lookup/ek3dj2stvq?authuser=0&hs=179 |
| ARM Processor | EC4C02 | B | https://meet.google.com/lookup/g6slr6j6we?authuser=0&hs=179 |
| ARM Processor Lab | EC4L02 | A | https://meet.google.com/lookup/ek3dj2stvq?authuser=0&hs=179 |
| ARM Processor Lab | EC4L02 | B | https://meet.google.com/lookup/g6slr6j6we?authuser=0&hs=179 |

**S Parameshwara**

**Data Transfer group of instructions related programs**

**; ARM ALP to transfer data b/n registers**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

ldr r0,=0x12345678 ; load operation

ldr r1,=0xabcdef01

ldr r2,=0xffffffff

ldr r3,=0x87654321

ldr r4,=0x01fedcba

mov r5,r0 ; data transfer

mov r6,r1

mov r7,r2

mov r8,r3

mov r9,r4

stop b stop

end

**; ARM ALP to data transfer b/n code. mem. into data.mem.**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

n equ 5

entry

export Reset\_Handler

Reset\_Handler

mov r0,#n ; n 32-bit data (counter)

ldr r1,=src

ldr r2,=dst

cntd ldr r3,[r1],#4

str r3,[r2],#4

subs r0,r0,#1

cmp r0,#0

bne cntd

stop b stop

src dcd 0x12345678, 0xabcdef01, 0x87654321, 0x1379ace1,0x98765432

area mydata,data,readwrite

dst space 0

end

**; ARM ALP to data transfer b/n code. mem. into data.mem. halfwordwise**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

n equ 5

entry

export Reset\_Handler

Reset\_Handler

mov r0,#n ; n 16-bit data (counter)

ldr r1,=src

ldr r2,=dst

cntd ldrh r3,[r1],#2

strh r3,[r2],#2

subs r0,r0,#1

cmp r0,#0

bne cntd

stop b stop

src dcw 0x1234, 0xabcd, 0x8765, 0x1379,0x9876

area mydata,data,readwrite

dst space 0

end

**; ARM ALP to data transfer b/n code. mem. into data.mem. bytewise**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

n equ 5

entry

export Reset\_Handler

Reset\_Handler

mov r0,#n ; n 8-bit data (counter)

ldr r1,=src

ldr r2,=dst

cntd ldrb r3,[r1],#1

strb r3,[r2],#1

subs r0,r0,#1

cmp r0,#0

bne cntd

stop b stop

src dcb 0x12, 0xab, 0x87, 0x13,0x98

area mydata,data,readwrite

dst space 0

end

**; ARM ALP to data transfer b/n code. mem. into data.mem. wordwise simultaneously**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

ldr r1,=src

ldr r2,=dst

ldmia r1,{r3-r7}

stmia r2,{r3-r7}

stop b stop

src dcd 0x12345678, 0xabcdef01, 0x87654321, 0x1379ace1,0x98765432

area mydata,data,readwrite

dst space 0

end

**; ARM ALP to data transfer b/n code. mem. into data.mem. bytewise random memory locations**

**; (1st, 4th data, 6th data, 7th data, 10th data)**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

ldr r1,=src

ldr r2,=dst

ldrb r3,[r1]

strb r3,[r2]

ldrb r3,[r1,#3]

strb r3,[r2,#3]

ldrb r3,[r1,#5]

strb r3,[r2,#5]

ldrb r3,[r1,#6]

strb r3,[r2,#6]

ldrb r3,[r1,#9]

strb r3,[r2,#9]

stop b stop

src dcb 0x12,0xab,0x87,0x13,0x98,0x18,0x14,0x15,0x16,0x17

area mydata,data,readwrite

dst space 0

end

**; ARM ALP to data transfer b/n code. mem. into data.mem. bytewise alternate memory locations**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

n equ 10

entry

export Reset\_Handler

Reset\_Handler

mov r0,#n ; n 8-bit data (counter)

ldr r1,=src

ldr r2,=dst

cntd ldrb r3,[r1,#2]!

strb r3,[r2,#2]!

subs r0,r0,#1

cmp r0,#0

bne cntd

stop b stop

src dcb 0,1,2,3,4,5,6,7,8,9,0x10,0x11,0x12,0x13,0x14,0x15,0x16,0x17,0x18,0x19,0x20

align

area mydata,data,readwrite

dst space 0

align

end

**; ARM ALP to data transfer b/n code. mem. into data.mem. bytewise using ldmia !**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

ldr r0,=src ; starting src address

ldr r1,=dst

ldr r2,=src+4 ; src ending address

cntd ldmia r0!, {r3-r7}

stmia r1!,{r3-r7}

cmp r0,r2

bne cntd

stop b stop

src dcb 1,2,3,4,5

align

area mydata,data,readwrite

dst space 0

align

end

**; ARM ALP to data transfer b/n code. mem. into data.mem. bytewise in reverse order**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

n equ 5

entry

export Reset\_Handler

Reset\_Handler

mov r0,#n ; n 8-bit data (counter)

ldr r1,=src

ldr r2,=dst+4

cntd ldrb r3,[r1],#1

strb r3,[r2],#-1

subs r0,r0,#1

cmp r0,#0

bne cntd

stop b stop

src dcb 1,2,3,4,5

area mydata,data,readwrite

dst space 0

end

**; ARM ALP to data transfer b/n data. mem. into data.mem.**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

n equ 5

entry

export Reset\_Handler

Reset\_Handler

mov r0,#n ; n 8-bit data (counter)

ldr r1,=data1

ldr r2,=data2+10

cntd ldrb r3,[r1],#1

strb r3,[r2],#1

subs r0,r0,#1

cmp r0,#0

bne cntd

stop b stop

area mydata1,data,readwrite

data1 space 0

area mydata2,data,readwrite

data2 space 0

end

**; ARM ALP to data transfer b/n registers into data.mem. wordwise**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

ldr r0,=0x12345678

ldr r1,=0xabcdef01

ldr r2,=0x87654321

ldr r3,=0x10fedcba

ldr r4,=dst

str r0,[r4,#4]

str r1,[r4,#12]

str r2,[r4,#18]

str r3,[r4,#23]

stop b stop

area mydata,data,readwrite

dst space 0

end

**; ARM ALP to data transfer b/n registers into data.mem. wordwise in sequence**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

ldr r0,=0x12345678

ldr r1,=0xabcdef01

ldr r2,=0x87654321

ldr r3,=0x10fedcba

ldr r4,=src

; str r0,[r4],#4

; str r1,[r4],#4

; str r2,[r4],#4

; str r3,[r4]

stmia r4,{r0-r3}

stop b stop

area mydata,data,readwrite

src dcb 1,2,3,4,5

end

**; ARM ALP to data transfer b/n code. mem. into data.mem. bytewise string**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

ldr r1,=src

ldr r2,=dst

cntd ldrb r3,[r1],#1

strb r3,[r2],#1

cmp r3,#0

bne cntd

stop b stop

src dcb "I am Parameshwara S",0

area mydata,data,readwrite

dst space 0

end

**; ARM ALP to exchange b/n code. mem. into data.mem. bytewise**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

n equ 5

entry

export Reset\_Handler

Reset\_Handler

mov r0,#n ; n 8-bit data (counter)

ldr r1,=data1

ldr r2,=data2

cntd ldrb r3,[r1]

ldrb r4,[r2]

strb r3,[r2],#1

strb r4,[r1],#1

subs r0,r0,#1

cmp r0,#0

bne cntd

stop b stop

area mydata1,data,readwrite

data1 space 5

area mydata2,data,readwrite

data2 space 5

end

**; ARM ALP to increment memory content and then transfer**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

n equ 5

entry

export Reset\_Handler

Reset\_Handler

mov r0,#n ; n 32-bit data (counter)

ldr r1,=src

ldr r2,=dst

cntd ldr r3,[r1],#4

add r3,r3,#1

str r3,[r2],#4

subs r0,r0,#1

cmp r0,#0

bne cntd

stop b stop

src dcd 0x1234567a, 0xabcdef0b, 0x8765432c, 0x1379aced,0x9876543e

area mydata,data,readwrite

dst space 0

end

**; ARM ALP to increment the data memory content**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

n equ 5

entry

export Reset\_Handler

Reset\_Handler

mov r0,#n ; n 32-bit data (counter)

ldr r1,=data1

cntd ldrb r3,[r1]

add r3,r3,#1

strb r3,[r1],#1

subs r0,r0,#1

cmp r0,#0

bne cntd

stop b stop

area mydata,data,readwrite

data1 space 0

end

**; ARM ALP to verify stack operation (ascending stack) using stmea and ldmea**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0x10001000; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

mov r0,#1

mov r1,#2

stmea sp!, {r0}

stmea sp!, {r1}

ldmea sp!, {r2}

ldmea sp!, {r3}

stop b stop

area mydata,data,readwrite

data1 space 0

end

**; ARM ALP to verify stack operation (descending stack) using push and pop**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0x10001000; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

mov r0,#1

mov r1,#2

push {r0}

push {r1}

pop {r2}

pop {r3}

stop b stop

area mydata,data,readwrite

data1 space 0

end

**; ARM ALP to verify stack operation (descending stack) using stmfd and ldmfd**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0x10001000; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

mov r0,#1

mov r1,#2

stmfd sp!,{r0}

stmfd sp!,{r1}

ldmfd sp!,{r2}

ldmfd sp!,{r3}

stop b stop

area mydata,data,readwrite

data1 space 0

end

**; ARM ALP to verify stack operation (ascending stack) using stmea and ldmea simultaneously**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0x10001000; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

ldr r0,=0x12345678

ldr r1,=0xabcdef01

stmea sp!, {r0,r1}

ldmea sp!, {r2,r3}

stop b stop

area mydata,data,readwrite

data1 space 0

end

**; ARM ALP to verify exchange operation in different addressing modes**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

; using register addressing mode

ldr r0,=0x12345678

ldr r1,=0x90abcdef

mov r2,r0

mov r0,r1

mov r1,r2

; using register addressing mode

ldr r5,=0x12345678

ldr r6,=0x90abcdef

ldr r3,=data1

str r5,[r3],#4

str r6,[r3]

ldr r5,[r3],#-4

ldr r6,[r3]

; using stack memory

ldr r7,=0x12345678

ldr r8,=0x90abcdef

push {r7}

push {r8}

pop {r7}

pop {r8}

; using ex-or operation

ldr r9,=0x12345678

ldr r10,=0x90abcdef

eor r9,r9,r10 ; a = a ex-or b

eor r10,r9,r10 ; b = a ex-or b

eor r9,r9,r10 ; a = a ex-or b

stop b stop

area mydata1,data,readwrite

data1 space 5

area mydata2,data,readwrite

data2 space 5

end

**; ARM ALP to data transfer b/n code. mem. into data.mem. bytewise string using subroutine**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0x10001000; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

ldr r1,=src

ldr r2,=dst

bl copy

stop b stop

copy ldrb r3,[r1],#1

strb r3,[r2],#1

cmp r3,#0

bne copy

bx lr ; mov pc,lr

src dcb "I am Parameshwara S",0

area mydata,data,readwrite

dst space 0

end

**; ARM ALP to generate 5 numbers using subroutine**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0x10001000; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

mov r0,#5

ldr r1,=data1

mov r2,#1

cont strb r2,[r1],#1

bl gen

subs r0,r0,#1

cmp r0,#0

bne cont

stop b stop

gen add r2,r2,#5

bx lr ; mov pc,lr

area mydata,data,readwrite

data1 space 0

end

**; ARM ALP to generate 5 numbers using nested subroutine**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0x10001000; initilization of stack memory

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

mov r0,#5

ldr r1,=data1

mov r2,#1

cont strb r2,[r1],#1

bl gen

cmp r0,#0

bne cont

stop b stop

gen push {lr} ; stmfd r13!,{r14}

add r2,r2,#5

bl comp

pop {lr} ; ldmfd r13!,{r14}

bx lr ; mov pc,lr

comp subs r0,r0,#1

bx lr

area mydata,data,readwrite

data1 space 0

end