Jeremiah Webb CEC 320 Workshop 7

AREA my\_fancy\_asm\_code, CODE, READONLY ; Define the program area

; Export functions defined in this file. These functions need to be declared

; in the file calling them.

EXPORT task10

EXPORT task11

EXPORT task12

EXPORT task13

EXPORT task14

EXPORT task15

IMPORT gPtrArray10a

IMPORT gPtrArray11a

IMPORT gPtrArray12a

IMPORT gPtrArray13a

IMPORT gPtrArray14a

IMPORT gPtrArray15a

IMPORT gVar1

ALIGN ; Align the data in the boundary of 4 bytes.

task10 PROC

LDR r0, =gPtrArray10a ; Loading the address of the global variable gPtrArray10a

LDR r0, [r0] ; Loading the content of the global variable gPtrArray10a

LDR r1, =gVar1 ; Loading the address of the global variable gVar1

LDR r1, [r1] ; Loading the content of the global variable gVar1

MOV r2, #0 ; variable (int) i

task10\_loop

CMP r2, r1 ; test = r2 - r1

BGE task10\_end ; if test >= 0, then branch to task10\_end

MOV r3, r2, LSL #2 ; r3 <- r2 \* 4

SUB r3, #15 ; r3 <- r3 - 15

STRB r3, [r0, r2] ; r3 -> mem[r0 + r2] or r3 -> mem[r0 + i]

ADD r2, #1 ; r2 <- r2 + 1

B task10\_loop ; branch to task10\_loop

task10\_end

BX lr ; return

ENDP

; If you need to use registers starting from r4, you need to PUSH them first to save the

; run-time environment for the caller. You need to POP them up at the exit of the code.

task11 PROC

LDR r0, =gPtrArray10a

LDR r0, [r0]

LDR r1, =gVar1

LDR r1, [r1]

MOV r2, #0

task11\_loop

CMP r2, r1 ; test = r2 - r1

BGE task11\_end ; if test >= 0, then branch to task10\_end

MOV r3, r2, LSL #5 ; r3 <- r2 \* 4

ADD r3, #2 ; r3 <- r3 - 15

STRB r3, [r0, r2] ; r3 -> mem[r0 + r2] or r3 -> mem[r0 + i]

ADD r2, #1 ; r2 <- r2 + 1

B task11\_loop ; branch to task10\_loop

task11\_end

BX lr

ENDP

task12 PROC

PUSH {r4-r5, lr}

LDR r0, =gPtrArray10a

LDR r0, [r0]

LDR r4, =gPtrArray12a

LDR r4, [r4]

LDR r1, =gVar1

LDR r1, [r1]

MOV r2, #0

task12\_loop

CMP r2, r1

BGE task12\_end

LDRSB r3, [r0, r2]

LDR r5, =10

SUB r5, r3

STRH r5, [r4, r2, LSL #1]

ADD r3, #1

STRB r3, [r0, r2]

ADD r2, #1

B task12\_loop

task12\_end

POP {r4-r5, pc} ; Pop lr to pc, which is the same as BX lr.

ENDP

task13 PROC

PUSH {r4-r5, lr}

LDR r0, =gPtrArray11a

LDR r0, [r0]

LDR r1, =gPtrArray13a

LDR r1, [r1]

LDR r2, =gVar1

LDR r2, [r2]

SUB r2, #1

MOV r3, #0

task13\_loop

CMP r3, r2

BGE task13\_end

LDRB r4, [r0]; load gPtr11a

LDRB r5, [r0, #1]; load gPtr11a + 1

ADD r5, r4; compute addition

STRH r5, [r1, r3, LSL #1]

ADD r3, #1; increment i

ADD r0, #1; increment gPtr11a

B task13\_loop

task13\_end

POP {r4-r5, pc}

ENDP

task14 PROC

PUSH {r4-r5, lr}

LDR r0, =gPtrArray12a

LDR r0, [r0]

LDR r4, =gPtrArray14a

LDR r4, [r4]

LDR r1, =gVar1

LDR r1, [r1]

SUB r1, #1

MOV r2, #0

task14\_loop

CMP r2, r1

BGE task14\_end

LDRSH r3, [r0]

LDRSH r5, [r0, #2]!

ADD r3, r5, LSL #3

STR r3, [r4, r2, LSL #2]

ADD r2, #1

B task14\_loop

task14\_end

POP {r4-r5, pc}

ENDP

task15 PROC

PUSH {r4-r5, lr}

LDR r0, =gPtrArray13a

LDR r0, [r0]

LDR r1, =gPtrArray15a

LDR r1, [r1]

LDR r2, =gVar1

LDR r2, [r2]

SUB r2, #1

MOV r3, #0; i = 0

task15\_loop

CMP r3, r2

BGE task15\_end

LDRH r4, [r0], #2; load gPtr13a to temp

LDRH r5, [r0]; load gPtr11a + 1

ADD r5, r4, r5, LSL # 4; = temp + 16 \* (\*gPtr13a)

STRH r5, [r1, r3, LSL #2]

ADD r3, #1; increment i

B task15\_loop

task15\_end

POP {r4-r5, pc}

ENDP

END