.section INTERRUPT\_VECTOR, "x"

.global \_start

\_start:

b \_Reset @posição 0x00 - Reset

ldr pc, \_undefined\_instruction @posição 0x04 - Intrução não-definida

ldr pc, \_software\_interrupt @posição 0x08 - Interrupção de Software

ldr pc, \_prefetch\_abort @posição 0x0C - Prefetch Abort

ldr pc, \_data\_abort @posição 0x10 - Data Abort

ldr pc, \_not\_used @posição 0x14 - Não utilizado

ldr pc, \_irq @posição 0x18 - Interrupção (IRQ)

ldr pc, \_fiq @posição 0x1C - Interrupção(FIQ)

\_undefined\_instruction: .word undefined\_instruction

\_software\_interrupt: .word software\_interrupt

\_prefetch\_abort: .word prefetch\_abort

\_data\_abort: .word data\_abort

\_not\_used: .word not\_used

\_irq: .word irq

\_fiq: .word fiq

INTPND: .word 0x10140000 @Interrupt status register

INTSEL: .word 0x1014000C @interrupt select register( 0 = irq, 1 = fiq)

INTEN: .word 0x10140010 @interrupt enable register

TIMER0L: .word 0x101E2000 @Timer 0 load register

TIMER0V: .word 0x101E2004 @Timer 0 value registers

TIMER0C: .word 0x101E2008 @timer 0 control register

TIMER0X: .word 0x101E200c @timer 0 interrupt clear register

\_Reset:

bl main

b .

undefined\_instruction:

b .

software\_interrupt:

b do\_software\_interrupt @vai para o handler

prefetch\_abort:

b .

data\_abort:

b .

not\_used:

b .

irq:

b do\_irq\_interrupt @vai para o handler de interrupções IRQ

fiq:

b .

do\_software\_interrupt: @Rotina de Interrupçãode software

SUB lr, lr, #4

STMFD sp!,{R0-R12,lr}

add r1, r2, r3 @r1 = r2 + r3

LDMFD sp!,{R0-R12,pc}^

save\_registers:

STMFD sp!,{lr}

STMFD sp!,{r0}

ADR lr, nproc

LDR r0, [lr]

CMP r0, #0

ADREQ lr, linhaA

ADRNE lr, linhaB

LDMFD sp!,{r0}

STMIA lr!, {R0-R12} @salva registradores normais

MOV r2, lr

MRS r0, CPSR

MRS r1, SPSR

STMIA r2!, {r1} @salva CPSR

ORR r1, r1, #0x80 @desabilita interrupções

LDMFD sp!,{lr}

LDMFD sp!,{r3}

MSR cpsr\_c, r1

STMIA r2!, {sp, lr} @salva registradores especiais

STMIA r2!, {r3} @salva PC

MSR cpsr\_c, r0

MOV pc, lr

recover\_registers:

STMFD sp!,{r0}

ADR lr, nproc

LDR r0, [lr]

CMP r0, #0

ADREQ lr, linhaA

ADRNE lr, linhaB

LDMFD sp!,{r0}

LDMIA lr!, {R0-R12} @recupera os registradores normais

STMFD sp!, {R0-R3}

MOV r2, lr

MRS r0, CPSR

LDMIA r2!, {r1} @recupera CPSR

MSR spsr\_c, r1

ORR r1, r1, #0x80 @desabilita interrupções

MSR cpsr\_c, r1

LDMIA r2!, {sp, lr} @recupera registradores especiais

MSR cpsr\_c, r0

MOV lr, r2

LDMFD sp!,{R0-R3}

LDMIA lr!,{pc}^ @recupera PC

handler\_timer:

STMFD sp!,{lr}

LDR r0, TIMER0X

MOV r1, #0x0

@Escreve no registrador TIMER0X para limpar o pedido de interrupção

STR r1, [r0]

@ Inserir código que sera executado na interrupção de timer aqui

@(chaveamento de processos, ou alternar LED por exemplo)

ADR r0, nproc

LDR r1, [r0]

EOR r1, r1, #1

STR r1, [r0]

LDMFD sp!,{lr}

mov pc, r14 @retorna

do\_irq\_interrupt: @Rotina de interrupções IRQ

SUB lr, lr, #4

STMFD sp!,{lr}

BL save\_registers

LDR r0, INTPND @Carrega o registrador de status de interrupção

LDR r0, [r0]

TST r0, #0x0010 @verifica se é uma interupção de timer

@vai para o rotina de tratamento da interupção de timer

BLNE handler\_timer

B recover\_registers

timer\_init:

LDR r0, INTEN

LDR r1,=0x10 @bit 4 for timer 0 interrupt enable

STR r1,[r0]

LDR r0, TIMER0C

LDR r1, [r0]

MOV r1, #0xA0 @enable timer module

STR r1, [r0]

LDR r0, TIMER0V

MOV r1, #0xff @setting timer value

STR r1,[r0]

mrs r0, cpsr

bic r0,r0, #0x80

msr cpsr\_c,r0 @enabling interrupts in the cpsr

mov pc, lr

sp\_init:

@init sp for interrupt mode

MRS r0, CPSR\_all

BIC r0, r0, #0x1f

ORR r0, r0, #0b10010

MSR CPSR\_all, r0

LDR r13, =0x3000

@init sp for supervisor mode

MRS r0, CPSR\_all

BIC r0, r0, #0x1f

ORR r0, r0, #0b10011

MSR CPSR\_all, r0

LDR sp, =stack\_top

mov pc, lr

init\_proc\_1:

LDR r0, =0x1000

ADR r1, proc\_print\_1

ADR r2, linhaA

MRS r3, CPSR\_all

BIC r3, r3, #0x9f

ORR r3, r3, #0b10011

STR r0,[r2, #14\*4]

STR r3,[r2, #13\*4]

STR r1,[r2, #16\*4]

mov pc, lr

init\_proc\_2:

LDR r0, =0x2000

ADR r1, proc\_print\_2

ADR r2, linhaB

MRS r3, CPSR\_all

BIC r3, r3, #0x9f

ORR r3, r3, #0b10011

STR r0,[r2, #14\*4]

STR r3,[r2, #13\*4]

STR r1,[r2, #16\*4]

mov pc, lr

proc\_print\_1:

LDR r1, =3000000

LOOP1\_print\_1:

MOV r0, #0

LOOP2\_print\_1:

ADD r0, r0, #1

CMP r0, r1

BNE LOOP2\_print\_1

bl print\_1

B LOOP1\_print\_1

proc\_print\_2:

LDR r1, =3000000

LOOP1\_print\_2:

MOV r0, #0

LOOP2\_print\_2:

ADD r0, r0, #1

CMP r0, r1

BNE LOOP2\_print\_2

bl print\_2

B LOOP1\_print\_2

main:

bl sp\_init @initialize stack pointers

bl init\_proc\_1

bl init\_proc\_2

bl timer\_init @initialize interrupts and timer 0

bl c\_entry

b proc\_print\_1

stop:

b stop

linhaA:

.skip 17\*4

linhaB:

.skip 17\*4

nproc:

.byte 0