h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi
FACHBEREICH INFORMATIK

RECHNERARCHITEKTUR

WS2024

Termin 5

Stack, Unterprogramme, Sektionen, Iteration, Rekursion, Cache

Name, Vorname	Matrikelnummer	Anmerkungen
Datum	Raster (z.B. Mi3x)	Testat/Datum

Legende: V:Vorbereitung, D: Durchführung, P: Protokoll/Dokumentation, T: Testat

Ziele:
Verständnis für STACK Befehle und deren Nutzung bei Unterprogrammen. Ziel ist es Programme mit möglichst geringer Codegröße zu implementieren, sowie der Umgang mit einem Debugger/Simulator zu festigen.

Arbeitsverzeichnis:
Kopieren Sie sich das Verzeichnis, welches Ihnen im Praktikum zur Verfügung gestellt wird, in Ihr persönliches Verzeichnis. Dort stehen Ihnen dann alle benötigten Dateien zur Verfügung.

Vorbereitung
Arbeiten Sie sich in folgende Befehle des ARM-Prozessors und in den ARM Procedure Call Standard (APCS) ein:

Instruktion	Bedeutung
STMFD R13, {R1-R4, LR} PUSH {R1-R4, LR}	Speichert die Registerwerte R1 bis R4 sowie LR (=R14) an die Adresse, die in R13 (=SP) steht als voll absteigender Stack
LDMFD R13, {R1-R4, PC} POP {R1-R4, PC}	Lädt den Speicherinhalt von der Adresse, die in R13 (=SP) steht in Form eines voll absteigenden Stacks in die Register R1 bis R4 sowie PC (=R15)

Aufgabe 1:
In folgenden Tabellen ist jeweils ein Speicherauszug gezeigt. Welche Werte stehen in den Registern nach Ausführung des Blocktransferbefehls?

Inhalt	Adresse
11	0x8014
10	0x8010
9	0x800C
8	0x8008
7	0x8004
6	0x8000
5	0x7FFC
4	0x7FF8
3	0x7FF4
2	0x7FF0

R9 = 0x8000

LDMDA R9, {R1, R2, R6}	LDMFD R9, {R1, R2, R6}	LDMDB R9, {R1, R2, R6}
R1:_____	R1:_____	R1:_____
R2:_____	R2:_____	R2:_____
R6:_____	R6:_____	R6:_____

Aufgabe 2:
Gegeben ist ein direkt abbildender Cache
In einer Cache-Line liegen 2 Wörter (mit je 4 Byte)
Wie viele Bits werden für die Offsets verwendet?

ByteOffset ____ Bits, WordOffset ____ Bits, Insgesamt ____ Bits

Mit welcher Bitfolge wird Byte 2 in Wort 1 adressiert? _____

Der Cache hat 4 Cachelines. Die Speicheradressen haben 8 Bit.
Überlegen Sie sich, welche Bits für TAG und Index verwendet werden, vergessen Sie dabei die oben bestimmten Offset-Bits nicht!

TAG: _____

Tragen Sie in der folgenden Tabelle in der Spalte Treffer jeweils HIT oder MISS für den Lesezugriff ein. Die Reihenfolge der Lesezugriffe ist in der ersten Spalte angegeben.

Geben Sie bei einem MISS den in diesem Schritt neu abgespeicherten Tag in der richtigen Cacheline an. Unveränderte Cachelines müssen Sie nicht erneut eintragen (außer für einen Treffer, siehe unten). Bei einem HIT markieren Sie in der entsprechenden Zeile die getroffene Cacheline zusätzlich, indem Sie den bereits gespeicherten Tag-Wert erneut eintragen und umkringeln / einkreisen (vgl. vorausgefüllte Zeilen in den Tabellen).

		Tag			
Adresse	Treffer	Index 00	Index 01	Index 10	Index 11
0100 1010	MISS		010		
0100 1011	HIT		010		
1100 1101					
0101 1011					
0100 1010					
0101 1110					
1011 0001					

Warum entsteht ein Hit beim zweiten Zugriff trotz unterschiedlicher Adressen?

_____

_____

Aufgabe 3:

Machen sie sich mit der Fibonacci Funktion vertraut. Die Fibonacci Funktion ist folgendermaßen definiert

X	Fibonacci(X)
X = 0	0
X = 1	1
X >= 2	Fibonacci(X-2) + Fibonacci(X-1)

Aufgabe 4:

Schreiben Sie ein Programm int fiborec(int) in ARM Assembler, welches die Fibonacci Funktion rekursiv berechnet. Beachten Sie die APCS Konvention. Dieses Programm muss Variablen auf den Stack speichern und ist keine Blattroutine.

```
Define fiborec(n):
if n==0:
        return 0
if n<=2:
        return 1
return fiborec(n-1)+fiborec(n-2)
```

```
Define fiboiter(n):
a = 0
b = 1
for zaehler = 1 ..n:
        neu = a+b
        a = b
        b = neu
return a
```

Aufgabe 5:

Schreiben Sie ein Programm
int fiboiter(int) in ARM Assembler, welches die Fibonacci Funktion iterativ berechnet. Beachten Sie die APCS Konvention. Dieses Programm kann man unter Benutzung von Scratch Registern (r0-r3) und ohne Stackzugriff als Blattroutine schreiben.

Aufgabe 6:

Welches der beiden Programme Aufgabe3 oder Aufgabe 4 liefert bei gleichem X schneller ein Ergebnis und warum?

Aufgabe 7:

Schreiben Sie ein Programm (rekursive Version) mit Cache. Speichern Sie die errechneten Werte in dem Array fiboliste.

```
fiboliste = leer

Define fiboritercache(n):
if fiboliste[n] bekannt :
        return fiboliste[n]
a = 0
b = 1
for zaehler = 1 ..n:
        neu = a+b
        a = b
        b = neu
return fiboliste[n] = a
```

```
fiboliste = leer

Define fiboreccache(n):
if fiboliste[n] bekannt :
        return fiboliste[n]
if n==0:
        return 0
if n==1:
        return 1
return fiboliste[n] = fiboreccache(n-1)+fiboreccache(n-2)
```

Aufgabe 8: (Zusatz 1)

Schreiben Sie ein Programm (iterative Version) mit Cache. Speichern Sie die errechneten Werte in dem Array fiboliste.

Aufgabe 9: (Zusatz 2)

Ändern Sie fiboitercache so ab, dass es wie bei fiboreccache für große n schon die fiboliste (Cache) füllt.

Bericht

Der Praktikumsbericht dient zu Ihrer Nachbearbeitung des Praktikums und wird stichprobenhaft überprüft. Er beinhaltet auch den zeilenweise kommentierten Quelltext. Und die Antworten zu den gestellten Fragen. Auch eine Diskussion der Ergebnisse kann deutlich zum Lernerfolg beitragen.

Als Hilfestellung erhalten Sie ein Programm, das die Fibonacci Funktionen aufruft, ein Assemblerrahmenprogramm für die Unterprogramme und ein Makefile.

mainfibo.c:

```
#include <stdio.h>

int fiborec(int n);
int fiboiter(int n);
int fiboreccache(int n);
int fiboitercache(int n);

int main()
{
	int n;
	for ( n= 0; n <= 10 ; n++ )
	{
		printf("Fiborec %d: %d\n", n, fiborec(n));
	}
	for ( n= 0; n <= 10 ; n++ )
	{
		printf("Fiboiter %d: %d\n", n, fiboiter(n));
	}
	for ( n= 0; n <= 10 ; n++ )
	{
		printf("Fiboreccache %d: %d\n", n, fiboreccache(n));
	}
	return 0;
	for ( n= 0; n <= 10 ; n++ )
	{
		printf("Fiboitercache %d: %d\n", n, fiboitercache(n));
	}
	return 0;
}
```

makefile:

```
CC = arm-elf-gcc
CFLAGS = -O0 -g -c
LDFAGS = ""
ASFLAGS = -g -c
OBJS = mainfibo.o fibo.o
NAME = testfibo

$(NAME).elf: $(OBJS)
	$(CC) $(LDFLAGS) $(OBJS) -o $(NAME).elf

%.o: %.s
	$(CC) $(ASFLAGS) $< -o $@

%.o: %.S
	$(CC) $(ASFLAGS) $< -o $@

%.o: %.c
	$(CC) $(CFLAGS) $< -o $@
```

fibo.S

```
@ Name:                 Matrikelnummer:         Datum:
        .file "fibo.S"
        .text
        .align 2
        .global fiborec @ nimmt das Symbol fiborec in die globale Symboltabelle auf
        .type fiborec, function
fiborec:
@ Code fuer fiborec
        ..
        ..
.Lfe1:
        .size fiborec,.Lfe1-fiborec

        .global fiboiter @ nimmt das Symbol fiboiter in die globale Symboltabelle auf
        .type fiboiter,function
fiboiter:
@ Code fuer fiboiter
        ..
        ..
.Lfe2:
        .size fiboiter,.Lfe2-fiboiter

        .global fiboreccache @ nimmt das Symbol fiboreccache in die globale Symboltabelle auf
        .type fiboreccache,function
fiboreccache:
@ Code fuer fiboreccache
        ..
        ..
.Lfe3:
        .size fiboreccache,.Lfe3-fiboreccache

        .global fiboitercache @ nimmt das Symbol fiboitercache in die globale Symboltabelle auf
        .type fiboitercache,function
fiboitercache:
@ Code fuer fiboitercache
        ..
        ..
.Lfe4:
        .size fiboitercache,.Lfe3-fiboitercache
        .data
        .align 2
fiboliste: .space 100*4, 0

.end    @ End of File
```