Содержание

Этап 1. Исследование архитектуры ARM

Ознакомление с историей развития архитектуры

Согласно варианту, архитектурой, подлежащей рассмотрению, является архитектура ARM разработчика ARM Limited. Расшифровка аббревиатуры – Advanced RISC (reduced instruction set computer) Machine.

Первая поставленная передо мной задача – познакомиться с историей версий выбранной архитектуры. Для этого была изучена статья «Список архитектур ARM» [1] Дерево версий представлено на Рис. 1 Дерево версий архитектуры ARM

Ознакомление с выбранной моделью архитектуры

Проанализировав актуальность версий, для дальнейшего рассмотрения была выбрана модель ARM11 36 JF-S (version – ARMv6). Для ознакомления с ней была изучена документация [2]. Были проанализированы следующие разделы:

- Chapter 1. Introduction. (Базовое знакомство с моделью).

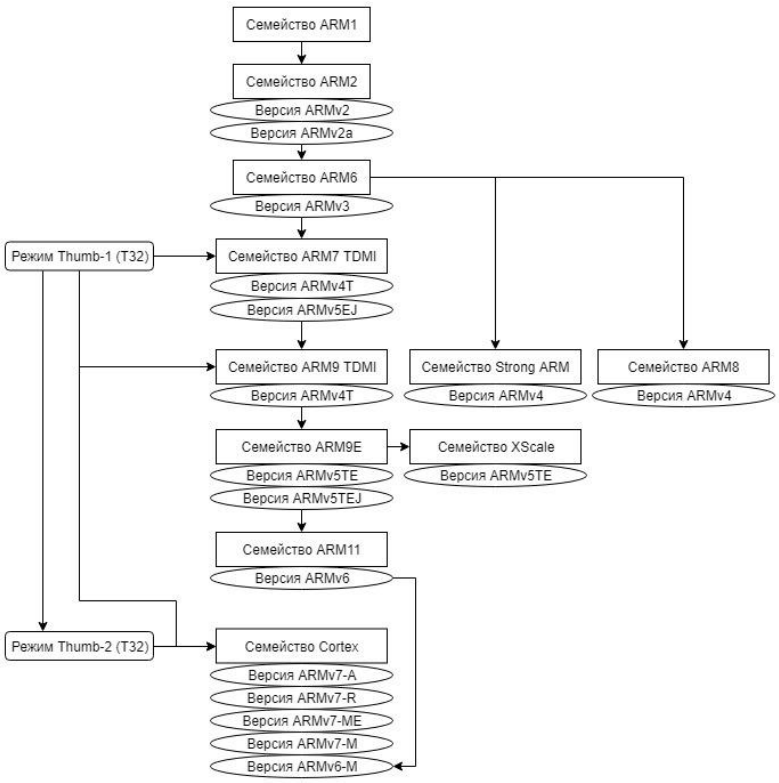

Рис. 1 Дерево версий архитектуры ARM

- Chapter 2. Programmer's model. (Общее представление о программной модели архитектуры, включая модель памяти, информацию о регистрах, а также о дополнительных инструкциях, характерных для модели).

При описании модели памяти было отмечено наличие 31 регистра общего назначения (разрядность – 32 бита), двух регистров состояния (CPSR и SPSR), хранящих текущее состояние процессора и зафиксированное при возникновении исключения, соответственно. Для модели характерно единое адресное пространство.

Следующим шагом изучения модели архитектуры является знакомство с набором инструкций. Для ознакомления была использована документация. [3] А именно следующие разделы:

- Chapter 2. ARM Instruction Set. Alphabetical list of ARM Instructions.
- Chapter 3. ARM Addressing Modes

По окончанию изучения материала был структурирован набор инструкций и составлена схема, изображенная на Рис. 2 Набор инструкций модели.

Data processing instructions												
Add and subtract	Compare	Logical	Byte-reverse	Parallel add and subtract		Sing or zero extend and add	Multiply and multiply-accumulate				Saturate	Miscellaneous
							32x32	32x16	16X16	Dual 16x16		
ADC	CMN	AND	REV	QADD16	QADD8	SADD16TO32	MLA	SMLAWy	SMLALxy	SMLAD	SSAT	CLZ
ADD	CMP	BIC	REV16	QADDSUBX	QSUB16	SADD8TO16	MUL	SMUWy	SMLAxy	SMLADX	SSAT16	NOP
QADD		CPY	REVSH	QSUBADDX	QSUB8	SADD8TO32	SMLAL SMMUL		SMULxy	SMLALD	USAT	PKHBT
QDADD		EOR		SADD16	SADD8	SUNPK16TO32	SMMLA SMMLAR			SMLALDX	USAT16	PKHTB
QDSUB		MOV		SADDSUBX	SHADD8	SUNPK8TO16	SMMLS			SMLSD		SEL
QSUB		MVN		SHADD16	SHSUB16	SUNPK8TO32	SMMLSR			SMLSDX		
RSB		ORR		SHADDSUBX	SHSUB8	UADD16TO32	SMMULR			SMLSLD		
RSC		TEQ		SHSUBADDX	SSUB8	UADD8TO16	SMULL			SMLSLDX		
SBC		TST		SSUB16	UADD8	UADD8TO32	UMLAL UMAAL			SMUAD		
SUB				SSUBADDX	UHADD8	UUNPK16TO32	UMULL			SMUADX		
				UADD16	UHSUB16	UUNPK8TO16				SMUSD		
				UADDSUBX	UHSUB8	UUNPK8TO32				SMUSDX		
				UHADD16	UQADD8							
				UHADDSUBX	UQSUB16							
				UHSUBADDX	UQSUB8							
				UQADD16	USAD8							
				UQADDUBX	USADA8							
				UQSUBADDX	USUB16							
				USUBADDX	USUB8							

Control instructions				
Branch	Status register handling	Change state	Software interrupt	Software breakpoint
B	MRS	CPS	SWI	BKPT
BL	MSR	SETEND		
BLX				
BX				
BXJ				

Load and store instructions						
Load	Load multiply	Preload	Store	Store multiply	Swap	Synchronization primitives
LDR	LDM	PLD	STR	STM	SWP	CLREX
LDRB			SRS			LDREX
LDRBT						LDREXB
LDRD						LDREXD
LDRH						LDREXH
LDRSB						STREX
LDRT						STREXB
LDRSH						STREXD
RFE						STREXH

Coprocessor instructions	
Coprocessor	Alternative coprocessor
CDP	CDP2
LDC	LDC2
MCR	MCR2
MCRR	MCRR2
MRC	MRC2
MRRC	MRRC2
STC	STC2

Рис. 2 Набор инструкций модели

Этап 2. Исследование инструментария для создания модели прикладной архитектуры процессора

Знакомство с документацией, представленной руководителем практики

Для реализации модели необходимы для изучения:

- Правила написания нотации
- Описание структуры файлов .asm
- Описание структуры файлов .bin

Знакомство с тестовым окружением, предоставленным преподавателем

Для анализа работы модели, которую необходимо разработать, руководителем было предоставлено тестовое окружение и поставлена задача познакомиться с возможными опциями и функционалом в целом.

Конкретно нас интересовали возможности проверки кода на валидность, ассемблирование .asm файла и дисассемблирование .bin файла.

Этап 3. Реализация модели

Часть 1. Описание регистров

```
storage R0_S[32];
storage R1_S[32];
storage R2_S[32];
storage R3_S[32];
storage R4_S[32];
storage R5_S[32];
storage R6_S[32];
storage R7_S[32];
storage R8_S[32];
storage R9_S[32];
storage R10_S[32];
storage R11_S[32];
storage R12_S[32];
storage R13_S[32];
storage R14_S[32];
storage R15_S[32];

storage CPSR_S[32];
storage SPRS_S[32];
```

```
view CPSR = CPSR_S;
view SPRS = SPRS_S;

view N = CPSR_S[31];
view Z = CPSR_S[30];
view C = CPSR_S[29];
view V = CPSR_S[28];
view Q = CPSR_S[27];
view DNM_1 = CPSR_S[25..26];
view J = CPSR_S[24];
view DNM_2 = CPSR_S[20..23];
view GE = CPSR_S[16..19];
view DNM_3 = CPSR_S[10..15];
view E = CPSR_S[9];
view A = CPSR_S[8];
view I = CPSR_S[7];
view F = CPSR_S[6];
view T = CPSR_S[5];
view M = CPSR_S[0..4];
```

Листинг 1 Пример конфигурации регистров модели и их представлений

Часть 2. Описание блоков памяти

Для данной архитектуры характерно единое адресное пространство.

```
| memory: range ram[0x00000000..0xFFFFFFFF]
```

Часть 3. Описание набора инструкций

```
encode imm7 field = immediate[7] data;
encode imm8 field = immediate[8] data;
encode imm12 field = immediate[12] data;
encode imm15 field = immediate[15] data;
encode reg32 field = register{
        PC = 1111,
        LR = 1110,
        SP = 1101,
        R12 = 1100,
        R11 = 1011,
        R10 = 1010,
        R9 = 1001,
        R8 = 1000,
        R7 = 0111,
        R6 = 0110,
        R5 = 0101,
        R4 = 0100,
        R3 = 0011,
        R2 = 0010,
        R1 = 0001,
        R0 = 0000
};
```

Листинг 2 Пример написания битовых паттернов

```
instruction and1 = {condition.al, 00, 1, 0000, one_bit_field.b, reg32 as Rn, reg32 as Rd, imm12 as shifter_operand};
instruction and2 = {condition.al, 00, 0, 0000, one_bit_field.b, reg32 as Rn, reg32 as Rd, sequence varopnd};

instruction qadd = {condition.al, 0001 0000, reg32 as Rn, reg32 as Rd, 0000 /*as SBZ*/, 0101, reg32 as Rm };
instruction qdadd = {condition.al, 0001 0100, reg32 as Rn, reg32 as Rd, 0000 /*as SBZ*/, 0101, reg32 as Rm };
instruction qdsub = {condition.al, 0001 0110, reg32 as Rn, reg32 as Rd, 0000 /*as SBZ*/, 0101, reg32 as Rm };
instruction qsub = {condition.al, 0001 0010, reg32 as Rn, reg32 as Rd, 0000 /*as SBZ*/, 0101, reg32 as Rm };

instruction rsb1 = {condition.al, 00, 1, 0011, one_bit_field.b, reg32 as Rn, reg32 as Rd, imm12 as shifter_operand };
```

Листинг 3 Пример кодирования полей в составе опкода

Часть 4. Описание мнемоник команд

```
format arg2_sh is "{1}, {2}, {3}";
format arg3 is "{1}, {2}, {3}";
format arg2_shifts is "{1}, {2}, {3}{4}";
```

Листинг 4 Пример описания форматных строк

```
mnemonic adc for adc1(Rn, Rd, shifter_operand) arg2_sh;
mnemonic adc for adc2(Rn, Rd, shift_off, Rm) arg2_shifts when immediateShifts,
                    (Rn, Rd, Rs, Rm) arg2_shifts when registerShift;
mnemonic add for add1(Rn, Rd, shifter_operand) arg2_sh;
mnemonic add for add2(Rn, Rd, shift_off, Rm) arg2_shifts when immediateShifts,
                    (Rn, Rd, Rs, Rm) arg2_shifts when registerShift;
```

Листинг 5 Пример описания мнемоник команд

Результаты

Тестирование реализованной модели: проверка валидности кода

```
Architecture: ARM1136JF
TaskID: e336455f-1cb2-40f7-99ec-a441fd66f42e

RESULTS:
Task creation time: 26.05.2019 21:55:55
Task state: Finished
Task finish time: 26.05.2019 21:55:55
Here is task output interpreted with UTF8:
Architecture 'ARM1136JF' seems to be valid for now.
```

Листинг 6 Результат проверки кода на валидность

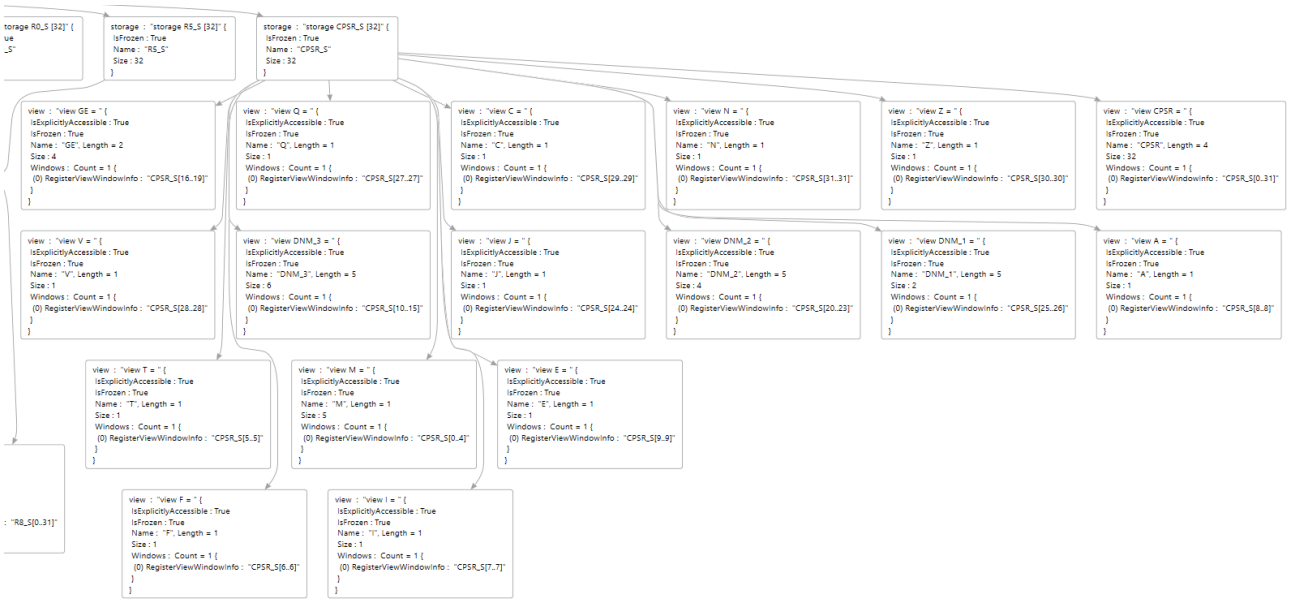

Рис. 3 Диаграмма представления регистра состояния CPSR

Тестирование реализованной модели: ассемблирование

```
adc PC,R7,0xFF7
add R1,R2,0xFF7
bl 0xFFFFFF
bic R0,R1,0xFF7
```

Листинг 7 Пример содержимого .asm файла

```
ARM1136JF
TaskID: 960a7cb0-f0dd-4d1b-8ba0-6e80bdaab326

RESULTS:
Task creation time: 26.05.2019 22:02:53
Task state: Finished
Task finish time: 26.05.2019 22:02:53
Here is task output interpreted with UTF8:
Architecture 'ARM1136JF' seems to be valid for now.
Assembling accomplished.
Binary module written.
```

Листинг 8 Результат ассемблирования файла

Тестирование реализованной модели: дисассемблирование

```
RESULTS:
Task creation time: 26.05.2019 22:05:26
Task state: Finished
Task finish time: 26.05.2019 22:05:26
Here is task output interpreted with UTF8:
Binary module disassembled.
```

Листинг 9 Результат дисассемблирования файла

Анализ результатов тестирования

В результате выполнения поставленных задач была реализована модель, которая по итогу проведения тестирования корректна (ассемблируется и дисассемблируется). Результат дисассемблирования (output.asm) идентичен исходным входным данным (input.asm), что говорит о ее работоспособности и применимости в других проектах. Результат дисассемблирования с использованием предоставленного тестового окружения совпадает с результатом дисассемблирования на сторонней машине.

Литература

1. Список архитектур ARM [Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/Список_архитектур_ARM
2. [Книга] ARM1136JF-S™ and ARM1136J-S™; Revision: r1p5; Technical Reference Manual
3. [Книга] ARM Architecture. Reference Manual Architecture. Reference Manual.