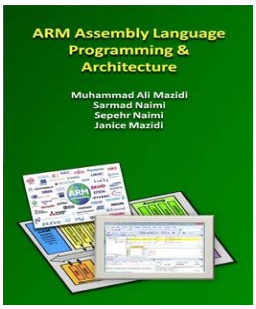
ARM Assembly Programming

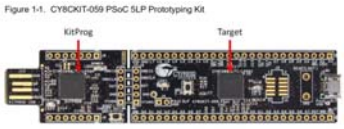
Textbook *ARM Assembly Language Programming & Architecture* by Mazidi, et al.



1. ARM and Microcontrollers
2. ARM Architecture and Assembly Language Programming
3. Arithmetic and Logic Instructions and Programs
4. Branch, Call, and Looping in ARM
5. Signed Integer Numbers Arithmetic
6. ARM Memory Map, Memory Access, and Stack
7. Floating-point expression

CPUlator Computer System Simulator ARM emulator – Coding Environment

PSoC® Creator™



1

EECE416 Microcontroller Fundamentals

i. Background:

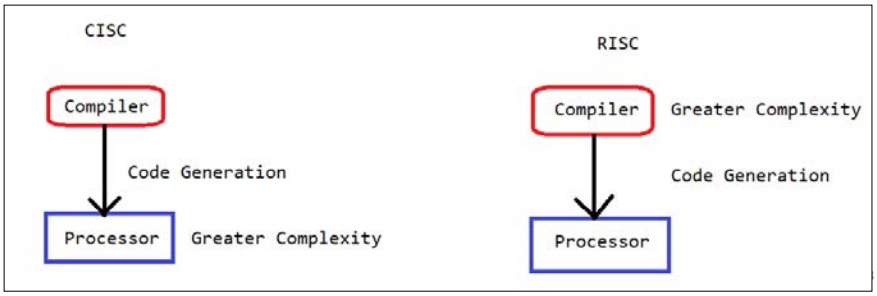
- Specific microprocessor core for embedded-systems and System-on-chip (SoC):
- ARM architecture:
- ARM processors are embedded in products:
- **Goal of the class**:

2

ii. ARM Embedded Systems

RISC Design Philosophy

- ARM core uses RISC (Reduced Instruction Set Computer) architecture
- cf. CISC (Complex Instruction Set Architecture) of Intel X86 architecture
- RISC Aim:
  - Delivering simple but powerful instructions → executes within a single cycle at a high clock speed
  - Reducing the complexity of instructions (performed by hardware) by providing flexibility and intelligence in software → greater demand on compiler.
  - Cf. CISC – relies more on hardware for instruction functionality and instructions are more complicated



ii. ARM Embedded Systems

Four (4) Major Design Rules of RISC Philosophy

1. Instructions
   1. Reduced number of instruction classes.
   2. Each instruction executes _____
   3. Programmer or compiler synthesizes _____
   4. Each instruction _____
2. Pipelines
   1. Instruction processing is broken down into smaller units
   2. executed in parallel by pipeline.
3. Registers
   1. RISC machines have _____
   2. Any register can contain _____
   3. Acts as the fast _____
4. Load-Store Architecture – for memory access
   1. RISC processor operates on data held in registers
   2. Cf. CISC – relies more on hardware for instruction functionality and instructions are more complicated

4

iii. ARM Design Philosophy

Physical Features which have driven ARM processor design

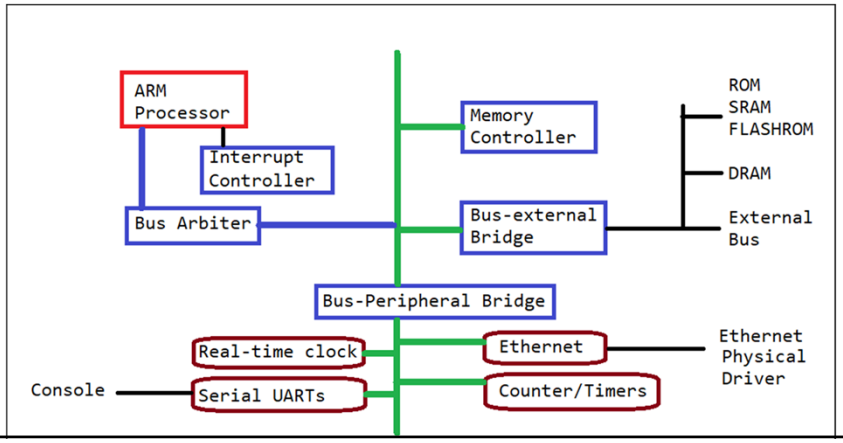
Portable embedded Systems (Target)

1. Battery power:
2. High code density:
3. Ability to use slow _____
4. Reduce the area of the die _____

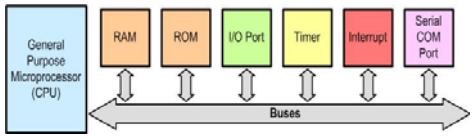
5

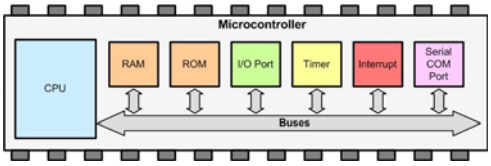
iv. Embedded System Hardware

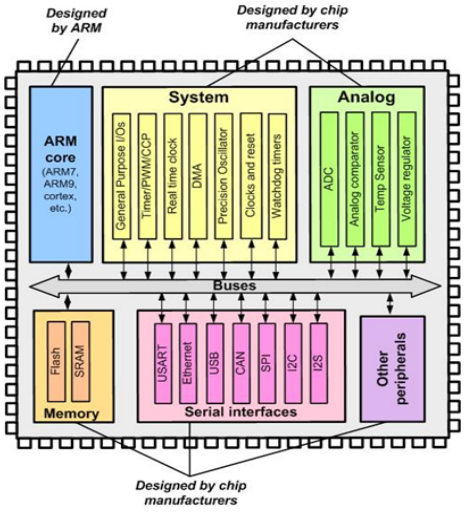
1. Embedded Systems – controls
2. Four (4) main hardware components of microcontroller
   1. The ARM Processor
   2. Controllers
   3. Peripherals
   4. Bus



iv. Embedded System Hardware - Packaging



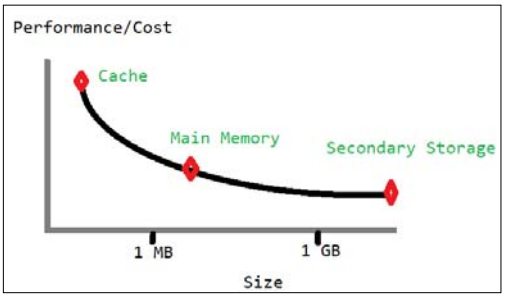




7

v. Memory and trade-off

1. Hierarchy (of off-chip memory)
   1. Storage trade-offs
   2. Cache:
   3. Secondary storage:
2. Width
   1. 8, 16, 32, or 64 bits
3. Types
   1. ROM
   2. Flash ROM –
   3. DRAM –
   4. SRAM –



8

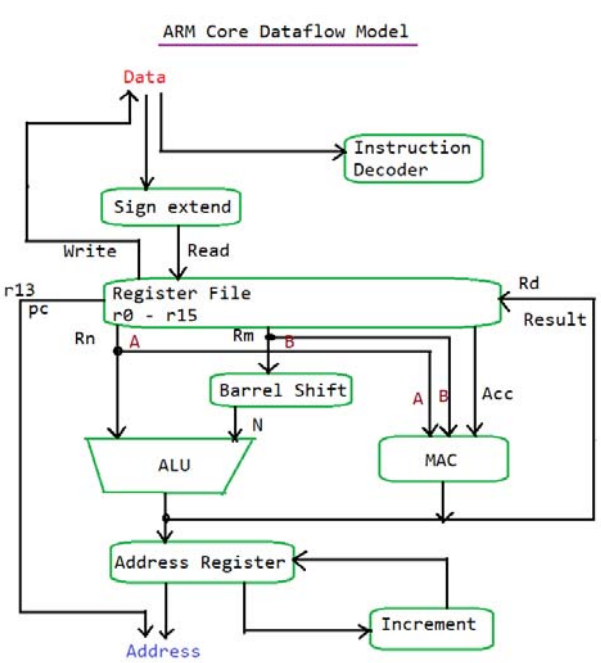
vi. Peripherals & Controllers

1. Peripherals
   1. Outside world an embedded system interacts with
   2. Input-output functions for the chip by connecting to other devices or sensors that are off-chip
   3. Range: From simple serial communication device to more complex 802.11 wireless device
   4. ARM peripherals are memory-mapped.
      1. Programming interface is a set of memory-addressed registers.

9

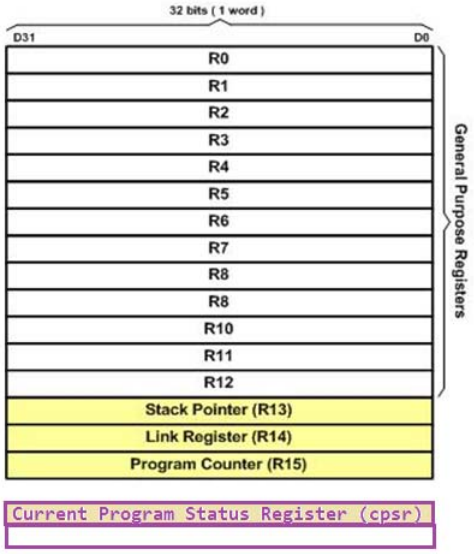
ARM Processor Fundamentals

1. Overview of the process core
2. Von Neumann (Cf. Harvard) Architecture
3. Rn & Rm – Source Reg
4. Rd – Destination Reg
5. MAC (Multiply-accumulate unit)
6. ALU (Arithmetic logic unit)
7. Load-Store Instruction: ALU generates address (of memory)
8. Rm can be pre-processed by Barrel Shifter



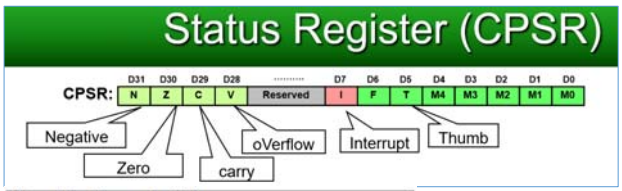
Registers

1. R0 – R12: General purpose registers hold either data or address
2. R13: stack pointer (**sp**) and stores the head of the stack in the current processor mode
3. R14: link register (**lr**) and is where the core puts the return address whenever it calls a subroutine
4. R15: program counter (**pc**) and contains the address of the next instruction to be fetched by the processor
5. Cpsr: (current program status register)



Cpsr (Current Program Status Register)

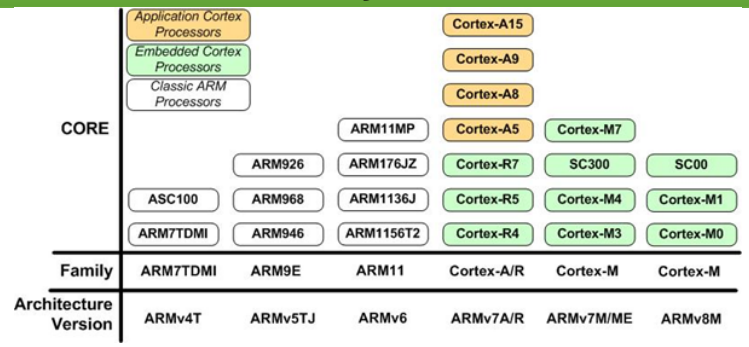
1. ARM uses **cpsr** to monitor and control internal operations
2. 32-bit register
3. Basic layout of a generic program status register



Flag	Flag Name	Set When
N	Negative	Bit 31 of the suresult is a binary 1
Z	Zero	The result is zero
C	Carry	The result causes an unsigned carry
V	oVerflow	The result causes an signed overflow

4. M4-M0: Processor modes
    1. Seven (7) processor modes including **user** (nonprivileged) mode – fast interrupt request; abort; interrupt request; supervisor; system; undefined; user.

ARM Core Family and Architecture



Revision	Example Core Implementation	ISA Enhancement
ARMv4T	ARM7DTMI, ARM9T	Thumb
ARMv5TE	ARM9E, ARM10E	Extra instructions added for changing state between ARM and Thumb
ARMv5TEJ	ARM7EJ, ARM926EJ	Java Acceleration
ARMv6	ARM11	Unaligned and mixed endian data handling; new multimedia instructions

13

Numbering System

Decimal numbers $1{,}245 = 1*10^3 + 2*10^2 + 4*10^1 + 5*10^0$

Binary Numbers 101011_2

$$1*2^5 + 0*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0 = 32 + 8 + 2 + 1 = 43_{10}$$

n	2^n	n	2^n	n	2^n	n	2^n
0	1	4	16	8	256	12	4096
1	2	5	32	9	512	13	8192
2	4	6	64	10	1024	14	16348
3	8	7	126	11	2048	15	32768

32-bit binary number

$2^{31} = 2.147484 \cdot 10^9$ $2^{32} = 4.294967 \cdot 10^9$

14

Numbering System

N-bit computer and accessible memory

2^{10}	Kilo		2^{30}	Giga		2^{50}	Penta
2^{20}	Mega		2^{40}	Tera		2^{60}	Exa

-3: Names for values of 2n, n = 10, 20, 30, 40, 50, 60

$2^{16} = 2^{10} * 2^{6} = 1K * 64 = 64K$

$2^{32} = 2^{30} * 2^{2} = 1G * 4 = 4G$

Conversion

64 32 16 8 4 2 1

1001010_2 = 1 0 0 1 0 1 0 = 64 + 8 + 2

433 - 256 = 177

2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	-	-	-	-	-	-	-	-

177 - 128 = 49

2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	1	-	-	-	-	-	-	-

15

Binary Numbering System

- Decimal to binary conversion:

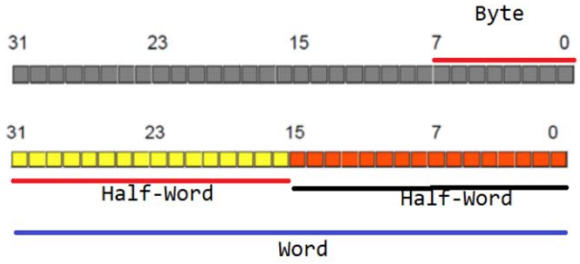
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 Power of 2

1 0 1 0 1 0 0 1 1 1 1 1

X_{15} X_{14} X_{13} X_{12} X_{11} X_{10} X_9 X_8 X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0

16

Numbering System



17

Practice

	Binary	Hexadecimal	Decimal
1.	100	_____	_____
2.	10101101	_____	_____
3.	1101110101	_____	_____
4.	11111011110	_____	_____
5.	10000000001	_____	_____
6.	_____	8EF	_____
7.	_____	10	_____
8.	_____	A52E	_____
9.	_____	70C	_____
10.	_____	6BD3	_____
11.	_____	_____	100
12.	_____	_____	527
13.	_____	_____	4128
14.	_____	_____	11947
15.	_____	_____	59020

18

Numbering System

Signed and Unsigned Numbers

Numbering System

- Pos -> Neg number Conversion

How to make out a negative number ?

⌘ Neg -> Pos number Conversion How to find the value of the number with sign=1 ?

The range of signed numbers in a given number of bits

$-2^{n-1} \sim (2^{n-1}-1)$ (... 0 ...) $0 \sim (2^n-1)$

	Signed Number Range	Unsigned Number Range
8-bit (Byte)	-128 ~ +127	0 ~ 255
16-bit (Word)	-32,768 ~ + 32,767	0 ~ 65,535
32-bit (DWord)	-2,147,483,648 ~ + 2,147,483,647	0 ~ 4,294,967,295

21

Practice for Signed Numbers

Answers in Hexadecimal number

⌘ Find the 8-bit expression for each of the following decimal numbers
- (a) 23
- (b) - 100

⌘ Find the 16-bit expression for each of the following decimal numbers
- (a) 15000
- (c) -923

⌘ Find the 32-bit expression for each of the following decimal numbers
- (a) 3874
- (b) - 100

22

Numbering System

Practice for Signed Numbers

- Find the decimal equivalent of each of the following 8-bit hexadecimal numbers
  - (a) 7C
  - (b) E1
- Find the decimal equivalent of each of the following 16-bit hexadecimal numbers
  - (a) 6F20
  - (c) B64A
- Find the decimal equivalent of each of the following ~~16~~ 32-bit hexadecimal numbers
  - (a) 98C2417D
  - (b) FFFFFF78

Numbering Systems

23

Character Representation

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Table 2-5: ASCII Table