Embedded Systems and Microprocessor Systems

Yuhui Shi

A7-1011

shiyh@sustc.edu.cn

Who Am I?

- Yuhui Shi
- Room Zhiyuan (南山智园）A7-1011
- Chair Professor, Department of Computer Science and Engineering
- shiyh@sustc.edu.cn

Who Are You?

- Departments?
- Interests?
- How much do you know about microprocessors?

ARMology

ARM1 (1985) -> ARM2(1987) -> ARMv2 -> ARM3
-> ARMv2a

ARM Ltd. (1990)

ARMv3(1990) -> ARM6 -> ARM7(1994) -> ARM8 -> ARM9 ->
ARMv4T -> ARMv5T -> ARM10 -> ARMv6 -> ARMv7 (Cortex-
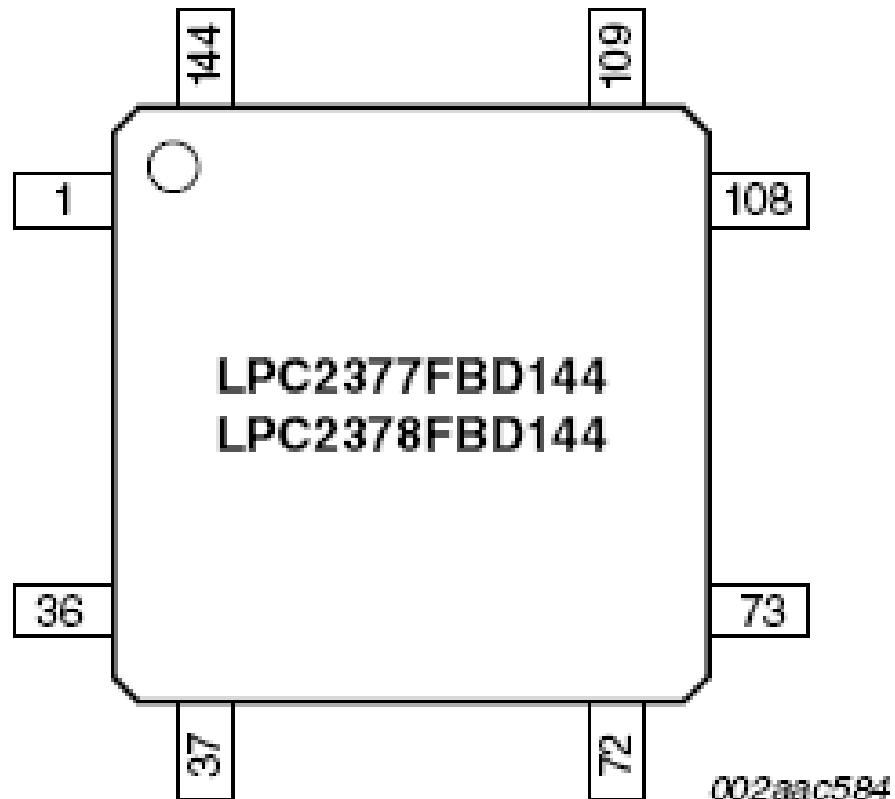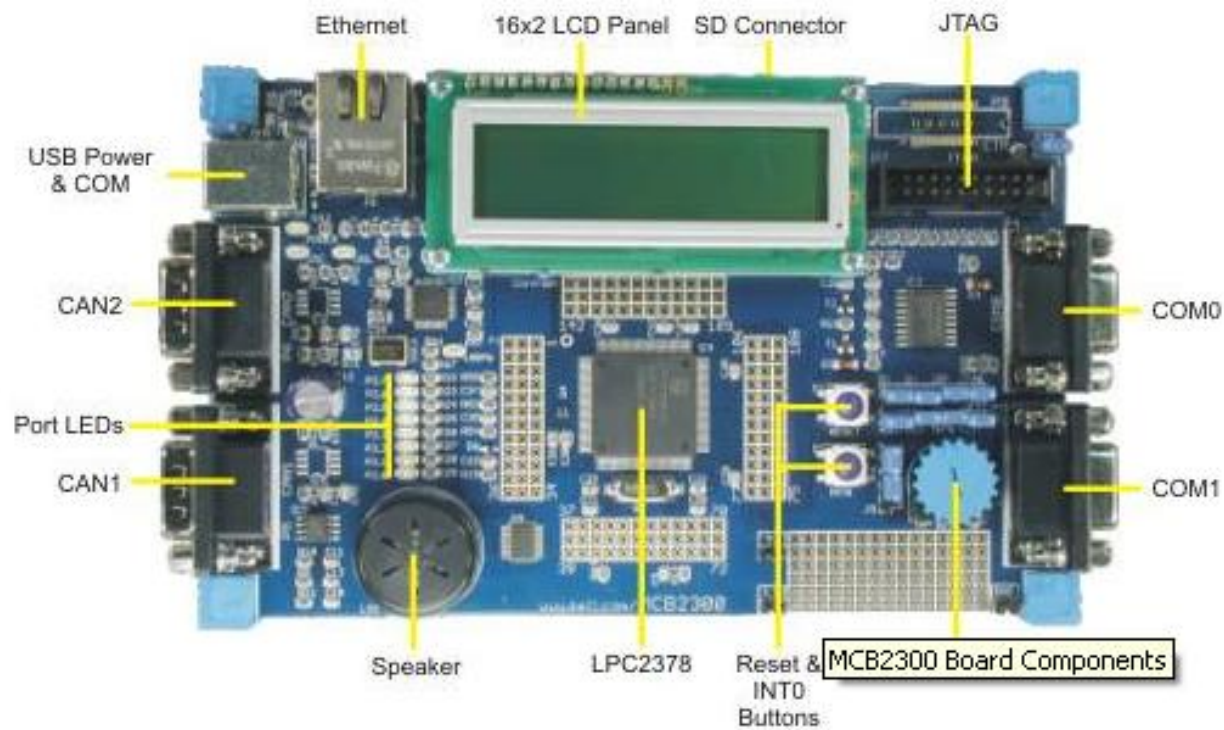M)(M3:2003) -> Cortex-A(2006) -> ARMv8(2011)

ARM7 ≠ ARMv7

Some companies (Intel, Marvell, Qualcomm, Microsoft, Apple,
Faraday and others) paid for 'architectural license' which
allows to design own cores.

ARM7TDMI landed nearly everywhere – MP3 players, cell phones, microwaves and any place where microcontroller could be used.

According to [ARM Ltd. page about ARM7](#) the ARM7 family is the world's most widely used 32-bit embedded processor family
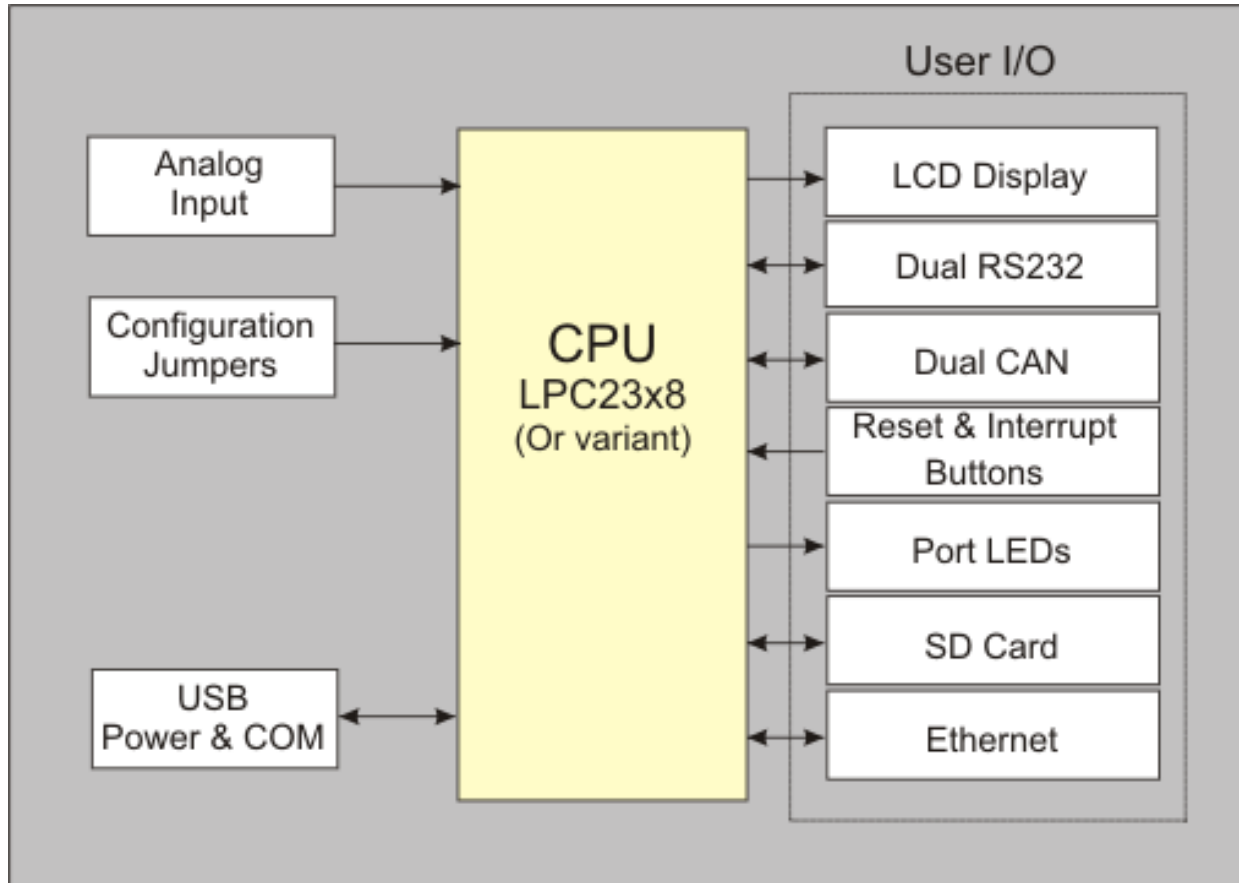
002aac584

Ethernet   16x2 LCD Panel   SD Connector   JTAG

USB Power & COM

CAN2

COM0

Port LEDs

CAN1

COM1

Speaker   LPC2378   Reset & INT0 Buttons

MCB2300 Board Components

002aac574

IC4 — LPC2378FBD144

JTAG / Reset section (left side):

Signal	Pin	Label
TRST_N	5	TRST_N
TDI	3	TDI
TMS	4	TMS
TCK	7	TCK
RTCK	143	RTCK
TDO	1	TDO
RST	24	NRESET
RSTOUTN	20	RSTOUTN
DBGEN		DBGEN
VDDA	6	DBGEN
	14	VDDA
VREFA	17	VREFA
DC-DC_3.3V	60	DC-DC_3.3V
DC-DC_3.3V	18	DC-DC_3.3V
VDD_9	138	VDD_9
VDD_7	114	VDD_7
VDD_6	102	VDD_6
VDD_4	77	VDD_4
VDD_3	62	VDD_3
VDD_1	41	VDD_1
CORE_3.3V	121	CORE_3.3V
VDD_1-9	98	CORE_1.8V
	81	CORE_1.8V
	21	CORE_1.8V
VSSA	15	VSSA
	119	CORE_VSS
	59	CORE_VSS
	22	CORE_VSS
	139	VSS_9
	117	VSS_7
	103	VSS_6
	79	VSS_4
	65	VSS_3
	44	VSS_1
VBAT	27	VBAT
ALARM	26	ALARM
RTCX1	23	RTCX1
RTCX2	25	RTCX2
X1	31	X1
X2	33	X2

Resistors: R55 10k, R53 0R, R54 0R, R14 0R, R15 0R, R63 0R
Capacitors: C34 100n, C35 100n, C5 100n, C15 100n, C36 100n, C37 100n, C38 100n, C39 100n, C40 100n, C41 100n, C42 100n
D6 BAT60A
Q4 32.768 kHz, C30 22p, C31 22p
Q3 12.000 MHz, C32 22p, C33 22p
+3.3V

Port 1 [0..15]:

Pin	Signal
136	P1.0/ENET-TXD0
135	P1.1/ENET-TXD1
133	P1.4/ENET-TX_EN
132	P1.8/ENET_CRS
131	P1.9/ENET_RXD0
129	P1.10/ENET_RXD1
128	P1.14/ENET-RX_ER
126	P1.15/ENET-RX_CLK

Port 1 [16..23]:

Pin	Signal
125	P1.16/ENET-MDC
123	P1.17/ENET-MDIO
46	P1.18/LED-AN/PWM1.1/CAP1.0
47	P1.19/TX_E-AN/PORT_PWR-AN/CAP1.1
49	P1.20/TX_DP-A/PWM1.2/SCK0
50	P1.21/TX_DM-A/PWM1.3/SSEL0
51	P1.22/RCV-A/PWRD-AN/MAT1.0
53	P1.23/RX_DP-A/PWM1.4/MISO0

Port 1 [24..31]:

Pin	Signal
54	P1.24/RX_DM-A/PWM1.5/MOSI0
56	P1.25/LOW_SP-A/HOST_EN-AN/MAT1.1
57	P1.26/SUSPEND-A/PWM1.6/CAP0.0
61	P1.27/INT-AN/OVR_CRNT-AN/CAP0.1
63	P1.28/SCL-A/PCAP1.0/MAT0.0
64	P1.29/SDA-A/PCAP1.1/MAT0.1
30	P1.30/PWRD-BN/VBUS-B/AD0.7
28	P1.31/OVR_CRNT-BN/SCK1/AD0.5

Port 2 [0..7]:

Pin	Signal
107	P2.0/PWM1.1/TXD1/TRACECLK
106	P2.1/PWM1.2/RXD1/PIPESTAT0
105	P2.2/PWM1.3/CTS1/PIPESTAT1
100	P2.3/PWM1.4/DCD1/PIPESTAT2
99	P2.4/PWM1.5/DSR1/TRACESYNC
97	P2.5/PWM1.6/DTR1/TRACEPKT0
96	P2.6/PCAP1.0/RI1/TRACEPKT1
95	P2.7/CAN_RX2/RTS1/TRACEPKT2

Port 2 [8..13]:

Pin	Signal
93	P2.8/CAN_TX2/TXD2/TRACEPKT3
92	P2.9/RXD2/EXTIN0
76	P2.10/EINT0N
75	P2.11/EINT1N/MCIDAT1/I2STX_CLK
73	P2.12/EINT2N/MCIDAT2/I2STX_WS
71	P2.13/EINT3N/MCIDAT3/I2STX_SDA

Port 3 [0..7]:

Pin	Signal
137	P3.0/D0
140	P3.1/D1
144	P3.2/D2
2	P3.3/D3
9	P3.4/D4
12	P3.5/D5
16	P3.6/D6
19	P3.7/D7

Port 3 [23..26]:

Pin	Signal
45	P3.23/D23
40	P3.24/D24
39	P3.25/D25
38	P3.26/D26

Port 0 [0..7]:

Pin	Signal	P0
66	P0.0/CAN_RX1/TXD3/SDA1	P0.0
67	P0.1/CAN_TX1/RXD3/SCL1	P0.1
141	P0.2/TXD0	P0.2
142	P0.3/RXD0	P0.3
116	P0.4/I2SRX_CLK/CAN_RX2/CAP2.0	P0.4
115	P0.5/I2SRX_WS/CAN_TX2/CAP2.1	P0.5
113	P0.6/I2SRX_SDA/SSEL1/MAT2.0	P0.6
112	P0.7/I2STX_CLK/SCK1/MAT2.1	P0.7

Port 0 [8..15]:

Pin	Signal	P0
111	P0.8/I2STX_WS/MISO1/MAT2.2	P0.8
109	P0.9/I2STX_SDA/MOSI1/MAT2.3	P0.9
69	P0.10/TXD2/SDA2/MAT3.0	P0.10
70	P0.11/RXD2/SCL2/MAT3.1	P0.11
29	P0.12/PORT_PWR-BN/MISO1/AD0.6	P0.12
32	P0.13/LED-BN/MOSI1/AD0.7	P0.13
48	P0.14/HOST_EN-BN/DEVICE_EN-BN/SSEL1	P0.14
89	P0.15/TXD1/SCK0/SCK	P0.15

Port 0 [16..23]:

Pin	Signal	P0
90	P0.16/RXD1/SSEL0/SSEL	P0.16
87	P0.17/CTS1/MISO0/MISO	P0.17
86	P0.18/DCD1/MOSI0/MOSI1	P0.18
85	P0.19/DSR1/MCICLK/SDA1	P0.19
83	P0.20/DTR1/MCICMD/SCL1	P0.20
82	P0.21/RI1/MCIPWR/CAN_RX1	P0.21
80	P0.22/RTS1/MCIDAT0/CAN_TX1	P0.22
13	P0.23/AD0.0/I2SRX_CLK/CAP3.0	P0.23

Port 0 [24..31]:

Pin	Signal	P0
11	P0.24/AD0.1/I2SRX_WS/CAP3.1	P0.24
10	P0.25/AD0.2/I2SRX_SDA/TXD3	P0.25
8	P0.26/AD0.3/AOUT/RXD3	P0.26
35	P0.27/SDA0	P0.27
34	P0.28/SCL0	P0.28
42	P0.29/USBD+A	P0.29
43	P0.30/USBD-A	P0.30
36	P0.31/USBD+B	P0.31
37	USBD-B	USBD-B

Port 4 [0..7]:

Pin	Signal	P4
52	P4.0/A0	P4.0
55	P4.1/A1	P4.1
58	P4.2/A2	P4.2
68	P4.3/A3	P4.3
72	P4.4/A4	P4.4
74	P4.5/A5	P4.5
78	P4.6/A6	P4.6
84	P4.7/A7	P4.7

Port 4 [8..15]:

Pin	Signal	P4
88	P4.8/A8	P4.8
91	P4.9/A9	P4.9
94	P4.10/A10	P4.10
101	P4.11/A11	P4.11
104	P4.12/A12	P4.12
108	P4.13/A13	P4.13
110	P4.14/A14	P4.14
120	P4.15/A15	P4.15

Port 4 [24..31]:

Pin	Signal	P4
127	P4.24/OEN	P4.24
124	P4.25/WEN	P4.25
118	P4.28/BLS2/MAT2.0/TXD3	P4.28
122	P4.29/BLS3/MAT2.1/RXD3	P4.29
130	P4.30/CSN0	P4.30
134	P4.31/CSN1	P4.31

KEIL SOFTWARE

Project-Version:	MCB2300-v3.1
Date / Time:	11.07.2007 17:42:13
Sheet:	1/6

KEIL SOFTWARE

Project-Version:	MCB2300-v3.1
Date / Time:	11.07.2007 17:42:13
Sheet:	5/6

KEIL SOFTWARE

Project-Version:	MCB2300-v3.1
Date / Time:	11.07.2007 17:42:13
Sheet:	2/6

KEIL SOFTWARE

Project-Version:	MCB2300-v3.1
Date / Time:	11.07.2007 17:42:13
Sheet:	3/6

KEIL SOFTWARE

Project-Version:	MCB2300-v3.1
Date / Time:	11.07.2007 17:42:13
Sheet:	4/6

What is a microprocessor?

- The part of a computer which does all the calculations is called the central processing unit (CPU).

- If a CPU is contained in one integrated circuit (silicon chip) it is called a microprocessor.

- The first microprocessor was the Intel 4004 which appeared in 1971. It was much less powerful than a 1970's computer because at that time integrated circuits only had a fraction of the number of transistors compared to modern integrated circuits

INTEL 4004
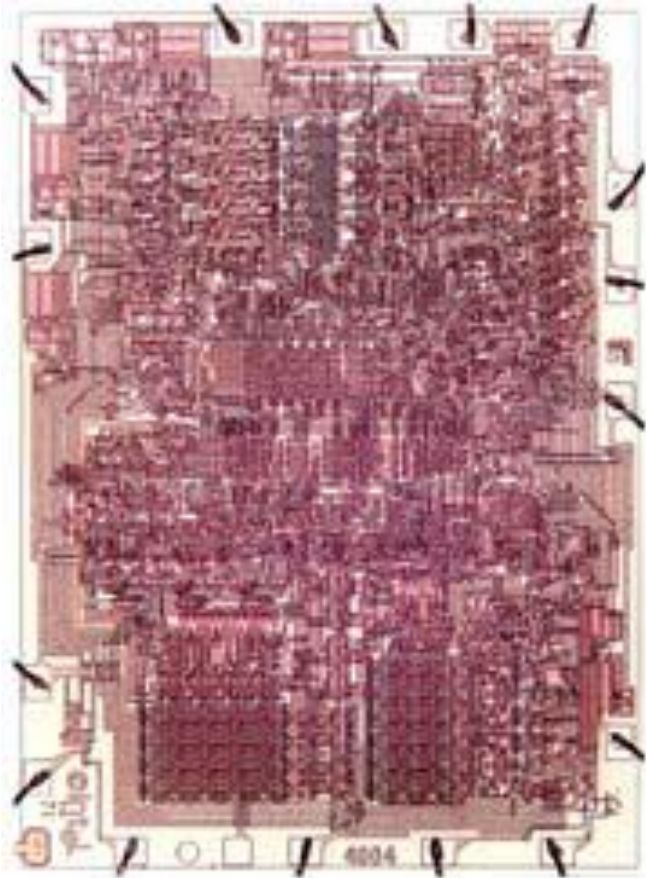
15 November 1971

2,300 transistors

10 micron (μm) gate length

4 bit address bus

100 kHz clock

Modern microprocessors

- With the advance of integrated circuit technology, microprocessors have become more sophisticated and are now much more powerful than a 1970's computer.

- Now an integrated circuit has enough transistors to contain several microprocessors or alternatively a microprocessor and associated circuits such as memory, analogue to digital converters (ADC), digital to analogue converters (DAC) etc.

INTEL Pentium
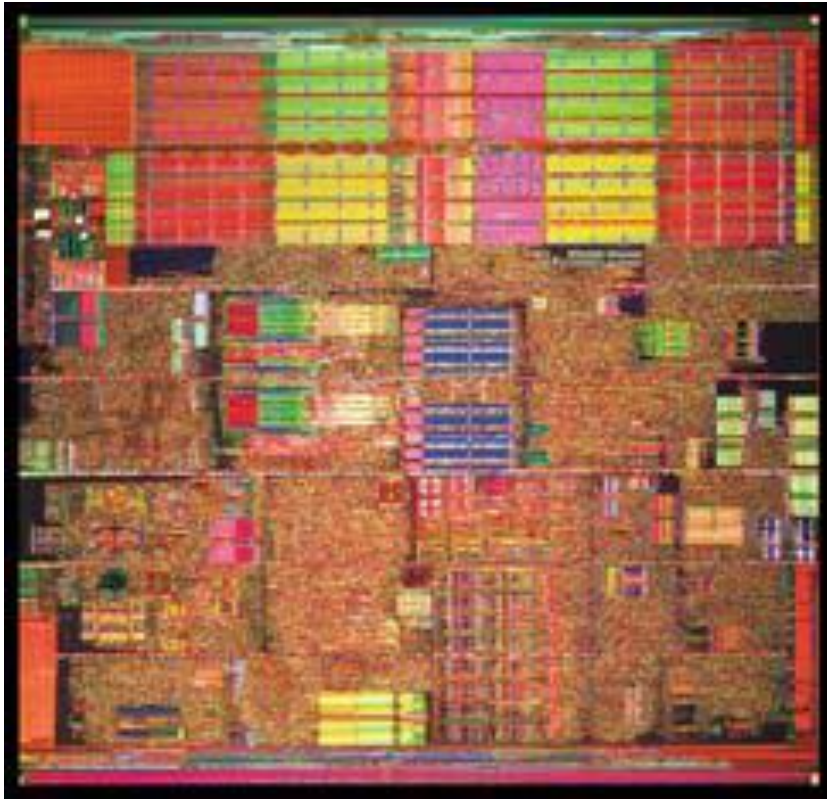
Feb. 2004 (0.09 μm)

125 million transistors

0.09 μm gate length

36 bit address bus

upto 3.6 GHz clock

112 mm² die area

min. power 73 Watts

The ARM microprocessor

- The ARM microprocessor is a example of a modern microprocessor which can be included (or embedded) in an integrated circuit containing many other circuits.

- It is designed by a British company, ARM Holdings PLC, which licenses the design to any integrated circuit manufacturer.

- All examples used in this course will refer to the ARM7 microprocessor where ARMv7 (Cortex-M3) is used in the laboratory exercises.

ARM7TDMI microprocessor

- 74,209 transistors
- 32 bit address bus
- 0.13 µm gate length
- upto 133 MHz clock
- 0.26 mm^2 die area
- max. power 8 mW

T: 16-bit **T**humb code
D: on-chip **D**ebug support
M: an enhanced **M**ultiplier
I: Embedded**I**CE hardward to give
   on-chip breakpoint and
   watchpoint support

Revision - binary

- All computers work on information and data coded in binary; that is base 2.

- The ARM microprocessor is a 32 bit processor so that numbers generally have 32 binary digits e.g.

  - 0011 1100 0100 0001 0101 0010 0100 1101$_2$
  - which in decimal (base 10) is 1,010,913,869$_{10}$

Revision - hexadecimal

- Because 32 bit binary numbers are very long, we generally use hexadecimal or base 16.

- Each hexadecimal digit is equivalent to 4 bits so a 32 bit number will have 8 hexadecimal digits.

- Conversion between hex and binary is simply a matter of substitution using the following table:

Converting hex to/from binary

Hex	Binary	Hex	Binary
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Converting hex to/from binary

- So the binary number:
  - $0011\ 1100\ 0100\ 0001\ 0101\ 0010\ 0100\ 1101_2$
  - converts to the following hexadecimal number:
    - $3C41524D_{16}$ or
    - 0x3C41524D
    - 0x denotes a hexadecimal number

Representing characters

- As well as manipulating numbers, computers also manipulate characters e.g.
  - in a word processing package like Microsoft Word.
  - Characters must be coded in binary before computers can process them
  - the standard coding for characters is ASCII (American Standard Code for Information Interchange).

ASCII

- ASCII is a 7 bit code and it codes for:
  - the standard 26 characters of English in both upper and lower case,
  - characters such as !#$%&()*+,- ./:;<=>?@[\]^_
  - the characters for numbers 0 to 9 and
  - control 'characters' such as line feed, carriage return, delete, escape, backspace.
  - It does not code for é or α or thousands of other non-English characters.

ASCII Table and Description

ASCII stands for American Standard Code for Information Interchange. Computers can only understand numbers, so an ASCII code is the numerical representation of a character such as 'a' or '@' or an action of some sort. ASCII was developed a long time ago and now the non-printing characters are rarely used for their original purpose. Below is the ASCII character table and this includes descriptions of the first 32 non-printing characters. ASCII was actually designed for use with teletypes and so the descriptions are somewhat obscure. If someone says they want your CV however in ASCII format, all this means is they want 'plain' text with no formatting such as tabs, bold or underscoring - the raw format that any computer can understand. This is usually so they can easily import the file into their own applications without issues. Notepad.exe creates ASCII text, or in MS Word you can save a file as 'text only'

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

Source: www.LookupTables.com

Extended ASCII Codes

As people gradually required computers to understand additional characters and non-printing characters the ASCII set became restrictive. As with most technology, it took a while to get a single standard for these extra characters and hence there are few varying 'extended' sets. The most popular is presented below.

128	Ç	144	É	161	í	177	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	163	ú	179	│	195	├	211	╙	227	π	243	≤
131	â	147	ô	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
134	å	150	û	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
136	ê	152	ÿ	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	·
137	ë	153	Ö	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	171	½	187	╗	203	╦	219	█	235	δ	251	√
139	ï	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	ⁿ
140	î	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	²
141	ì	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	
143	Å	160	á	176	░	192	└	208	╨	224	α	240	≡		

Source: www.LookupTables.com

Question

- The banner at the bottom of this page is ASCII code - what does it say?
  - Work on it for 2 minutes

3C 41 52 4D 2E 50 4F 57 45 52 45 44 3E 0D 0A

Computer as a processor

- A computer processes data to provide information e.g.
  - a computer in a supermarket which controls the check-out tills.
  - When the bar code of a bag of sugar is passed over the bar code reader the data it contains is processed by the computer.
  - The computer provides information such as price and description to show on the till display and to print on the receipt.

Computer versus human

- A human can perform the same functions as a computer e.g.
  - the human could read the number next to the bar code,
  - look that number up in a table,
  - read out the price and description to the customer and write them down on the receipt.
- The only difference is computers can do this much more quickly and they don't make mistakes because they get tired.

Instructions

- For a human to perform a task he/she needs instructions e.g.
  - to knit a jumper
    - the instructions are the knitting pattern
  - to bake a cake
    - the instructions are in the recipe.
- Computers also need instructions
  - so that the data is processed into information correctly.

Example - video game

- E.g. a video game
  - has data in the form of
    - joystick movement,
    - button presses
  - provides information in
    - pictures and
    - sounds.
  - The instructions are stored in the machine.

Joystick → COMPUTER → Pictures
Buttons → COMPUTER → Sounds

Example - automatic pilot

E.g. an automatic pilot ('fly by wire')

- has data on
  - aircraft speed,
  - height,
  - direction and
  - course.

Speed

Height

Direction

Course

COMPUTER

Engines
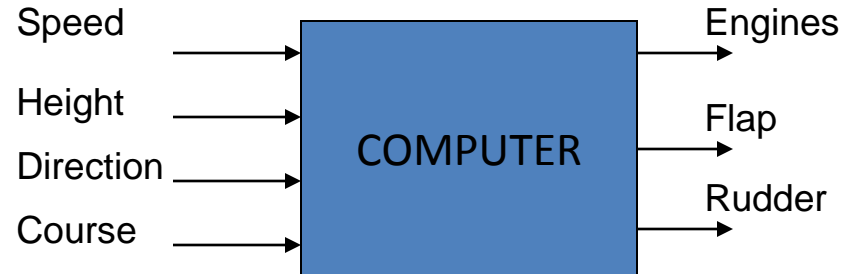
Flap

Rudder

- provides information to
  - control the engines,
  - flaps and
  - rudder.
- Again the computer has stored instructions so that it does this correctly.

A simple computer

- So a simple computer needs to be able to receive data, process it and return information back.

- In addition the computer must be able to store instructions.

- A keyboard can receive data into the computer.

- A CPU (central processing unit) can process it.

- A VDU (visual display unit) can return the information to the user.

- And computer memory can store instructions.

A simple computer architecture

- A simple computer layout or 'architecture' could be as shown below. Data and information pass between the blocks as electrical signals.
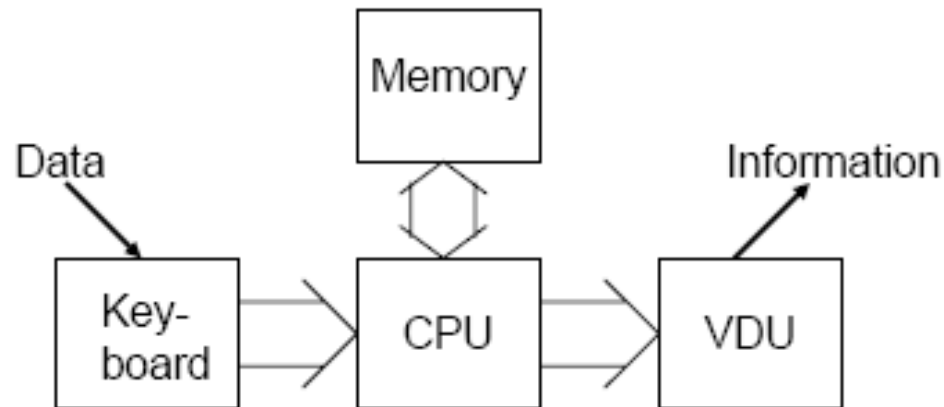
Architecture - is simplest best?

- Early computers (1950's) were laid out as shown on the previous slide.

- However this architecture can become very complicated if
  - more than one input device or
  - more than one output device needs to be connected
  - e.g. for the autopilot.

- If a CPU was connected to
  - 4 input devices,
  - 3 output devices &
  - 5 different memory devices
  - then it would need 12 connections using this architecture - very problematical.

The Bus Architecture

- The bus architecture can be extended over any number of devices. All devices have only one connection onto the 'bus'.

Data

Information

Key-board | CPU | Memory | VDU
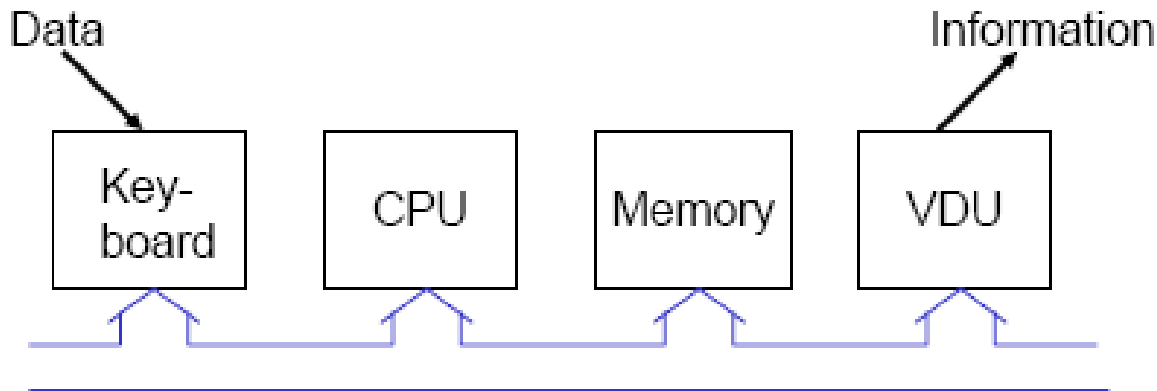
The Bus Architecture

A bus is a collection of electrical connections –

- normally 8, 16, 32 or 64 individual wires.
- 32 bits of data and information can pass along a 32 bit bus at the same time e.g.
  - the codes of 4 ASCII characters could go from CPU to VDU.
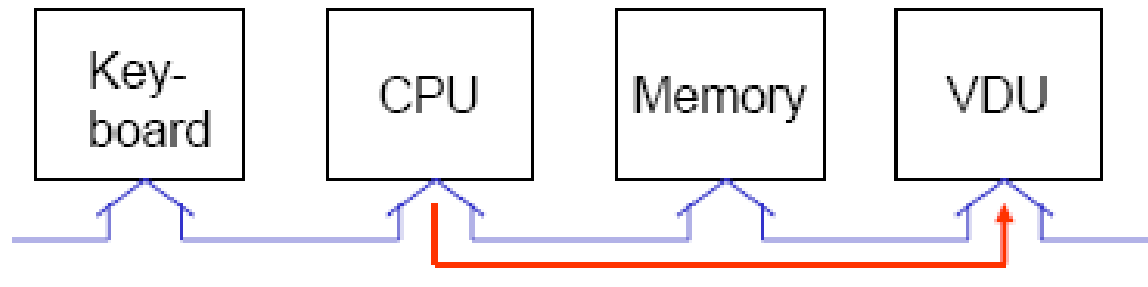
Controlling the Bus

- It is important that signals do not collide on the bus - only one device at a time can send data.

- The CPU controls all movements on the bus using special wires to activate devices and to synchronize the sending and receiving devices.

- These special connections are known as the control bus and they are completely independent of the bus along which data is passed.

- To avoid confusion, this is called the data bus.

A Third Bus

- In addition to the data bus and control bus there is a third bus called the address bus.

- The address bus is used by the CPU to determine which location in memory is sending or receiving data.
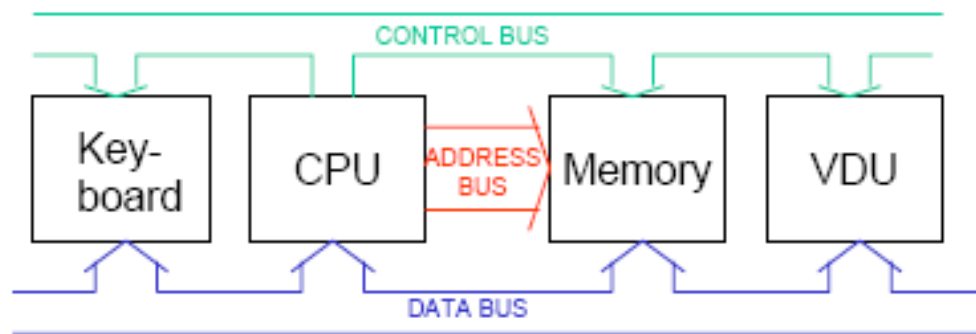
What is in memory?

- Memory is used to store the instructions which the CPU uses to process the data.

- Memory can also be used to store data in the form of numbers or characters.

- All computer memories work in binary so that instructions and data must be coded in binary e.g.
  - characters can be coded in ASCII
  - Instructions are coded in 'machine code'.

What is computer memory?

- Computer memory is a very big sequential logic circuit made up of thousands or millions of simple logic gates, such as a D type latch, which can remember a 0 or a 1, that is one bit of data.

- Groups of these gates are collected together in a memory 'location'.

- There are typically 8 bits of data in one location.

- Each memory location has a unique memory address.

Memory organization.

- Taking the ARM7TDMI microprocessor as an example
  - at each memory location
  - it has 8 bits of data
    - 8 bits is known as a byte.
- The ARM is a 32 bit processor and addresses are 32 bits long from 0x00000000 to 0xFFFFFFFF.
  - That means there can be up to 4,294,967,296 (or 2^{32}) different memory locations all with a unique memory address.
- In practice not all addresses are used for memory.

Some definitions.

- A byte is equal to 8 bits.

- A kilobyte is equal to 1024 bytes ($1024 = 2^{10}$).

- A megabyte (MB) is equal to 1024 kilobytes or 1048576 bytes ($1048576 = 2^{20}$).

- A gigabyte (GB) is equal to 1024 megabytes or 1073741824 bytes ($1073741824 = 2^{30}$).

- A 32 bit processor could be directly connected to 4GB of memory using a 32 bit address bus if every memory address had memory connected.

Some more definitions.

- Another term which is commonly used is a 'word'.

- The 'word' depends upon the processor used
  - for a 32 bit processor like the ARM
    - a word is equal to 32 bits or 4 bytes.
    - Similarly a 'half word' is 16 bits or 2 bytes.
  - Another way to say this is that the 'word length' is 32.

Questions

- For the ARM processor how many words are there in a kilobyte?

- How many kilobytes are there in a gigabyte?

- Work at it for 4 minutes.

The Central Processing Unit

- The central processing unit or CPU is the part of a microprocessor system that does all the 'work'.

- It interprets the instructions stored in memory.

- It performs the calculations.

- It controls the flow of data along the data bus.

- It determines which memory address to use.

Inside the CPU

- The CPU is designed to perform all of these functions as efficiently as possible.

- It can be subdivided into a number of blocks; each with a distinct function.

- Every CPU is different and we will concentrate on the ARM7 'core' - the CPU for the ARM7 range of microprocessors.

The ARM7 core

- The basic building blocks of the ARM7 core are:

Address register	Control unit
Register bank 16 x 32 bits	Instruction decoder
Arithmetic and logic unit - ALU	Instruction register
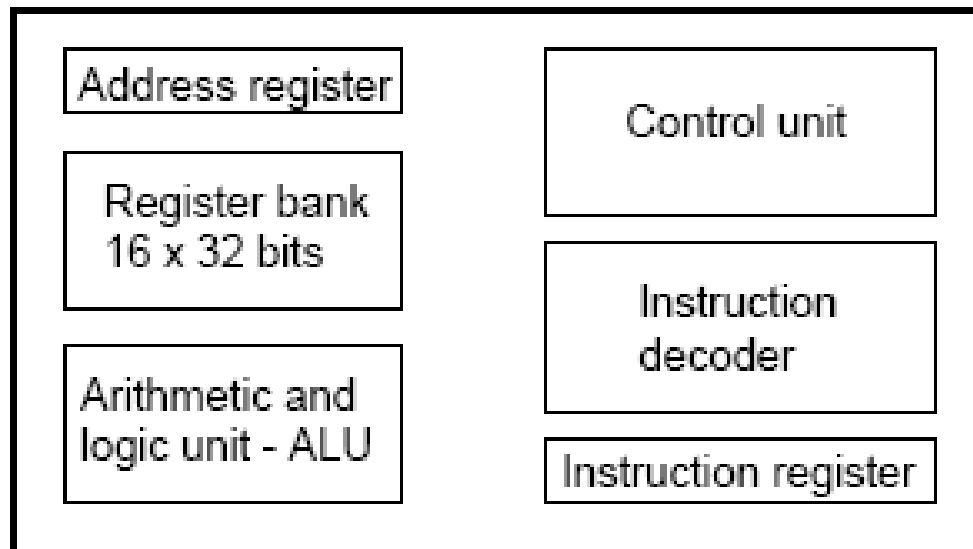
The instruction register

- The instructions stored in memory travel along the data bus to the CPU where they are loaded into the instruction register.

- ARM7 instructions are 32 bits long so the instruction register is a 32 bit memory device - not part of the main memory.

- The process of loading the instruction register from memory is known as a 'fetch'.
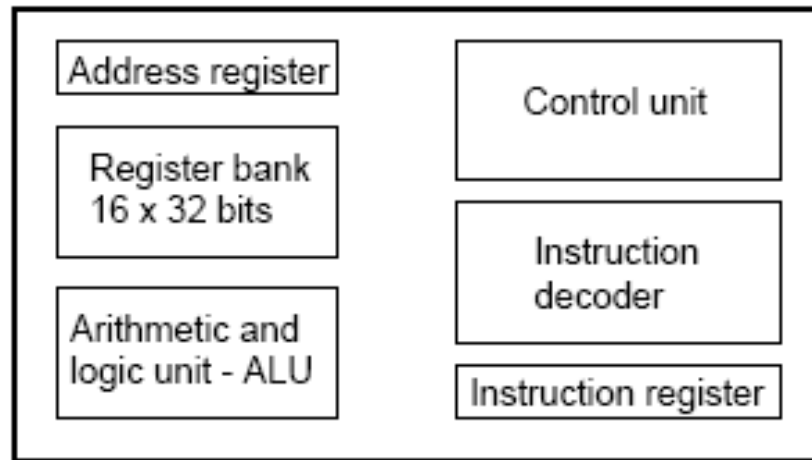
Instruction decoder, control unit

- The instructions are in 'machine code' and the instruction decoder determines the function of each instruction.

- The instruction decoder and control unit determine what the other parts of the CPU do.

- The control unit is also in charge of the control bus.

- The process of interpreting each instruction is known as the 'decode' cycle.

Address register

Register bank
16 x 32 bits

Arithmetic and
logic unit - ALU

Control unit

Instruction
decoder

Instruction register
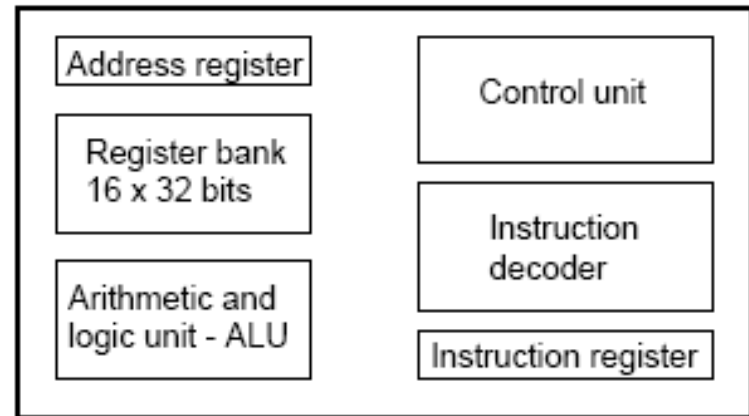
Arithmetic and logic unit

- The arithmetic and logic unit or ALU performs the mathematical functions as required.

- These may be arithmetic such as add, subtract or multiply or logical such as AND, OR, XOR etc.

- The process of performing each instruction is known as the 'execute' cycle.


```
Address register

Register bank
16 x 32 bits

Arithmetic and
logic unit - ALU

Control unit

Instruction
decoder

Instruction register
```
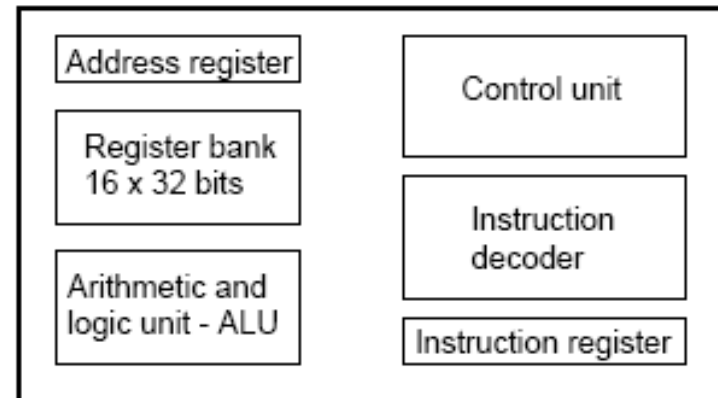
Register bank

- The register bank is a local memory for the CPU. It has 16 locations - each location can hold 32 bits of data.

- The registers are named r0, r1, r2, r3, ... etc. up to r15.

- They are used to hold data which is processed by the ALU and also hold the results of any calculation.

- Registers r13, r14 and r15 have special functions which we will cover later.

Address register	Control unit
Register bank 16 x 32 bits	Instruction decoder
Arithmetic and logic unit - ALU	Instruction register
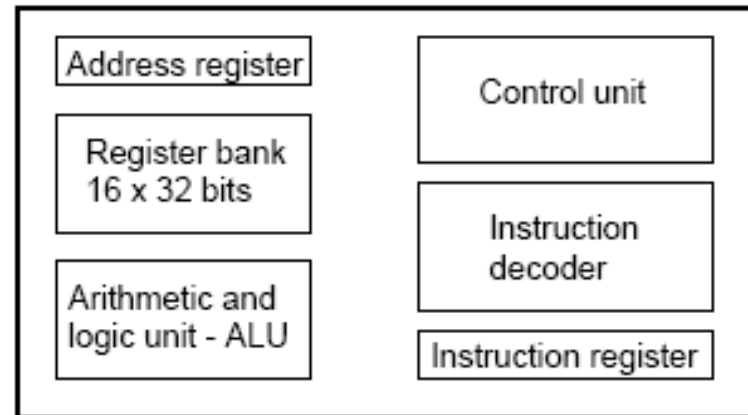
Address register

- The address register is a 32 bit memory device which holds a memory address value.

- Either this address may be for the memory location of the next instruction during the 'fetch' cycle.

- Or during the 'execute' cycle the address is for a memory location either containing data to be loaded into a register or where data from a register is to be stored.

Address register

Register bank
16 x 32 bits

Arithmetic and
logic unit - ALU

Control unit

Instruction
decoder

Instruction register
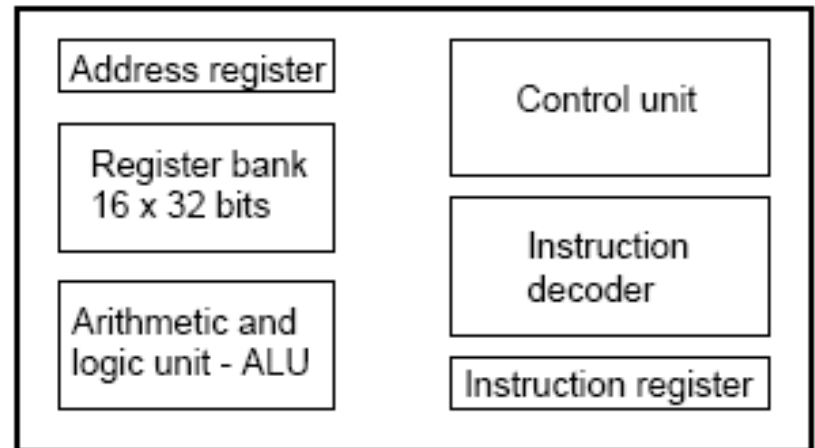
Fetch, decode, execute.

- The CPU performs three cycles sequentially.
  - During the fetch cycle an instruction in memory is loaded into the instruction register.
  - During the decode cycle the instruction is interpreted by the instruction decoder.
  - During the execute cycle
    - either the ALU performs a calculation on values held in registers or
    - a value in a register is stored into memory or
    - a value in memory is loaded into a register.

Internal connections

Internal connections
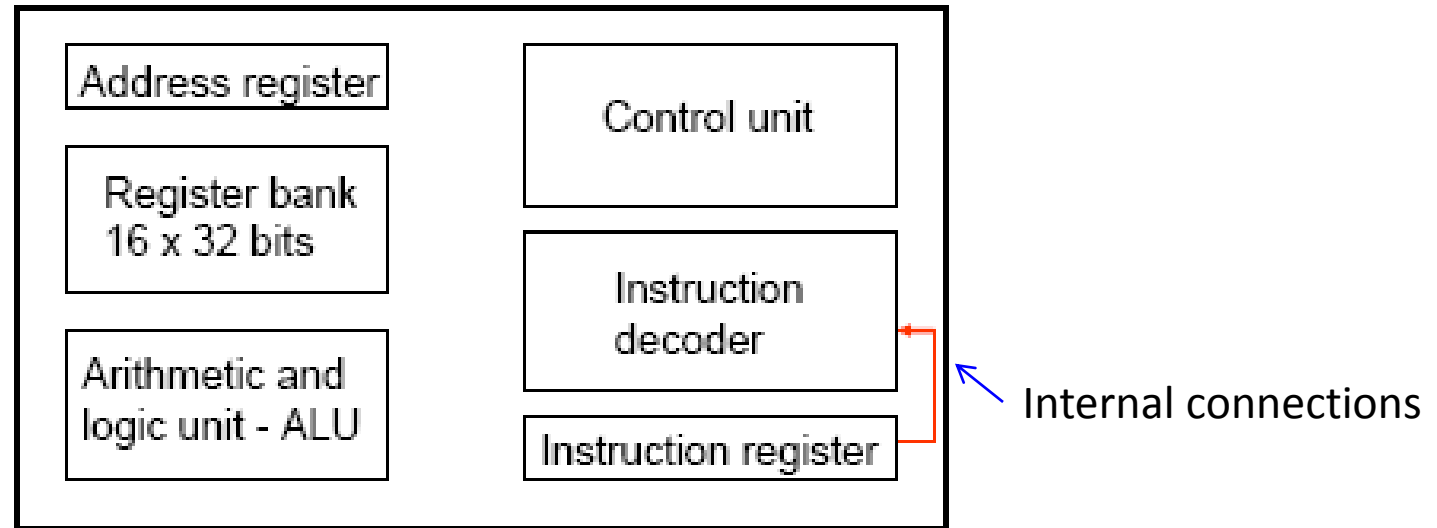
- The instruction register is connected to the instruction decoder.

Internal connections
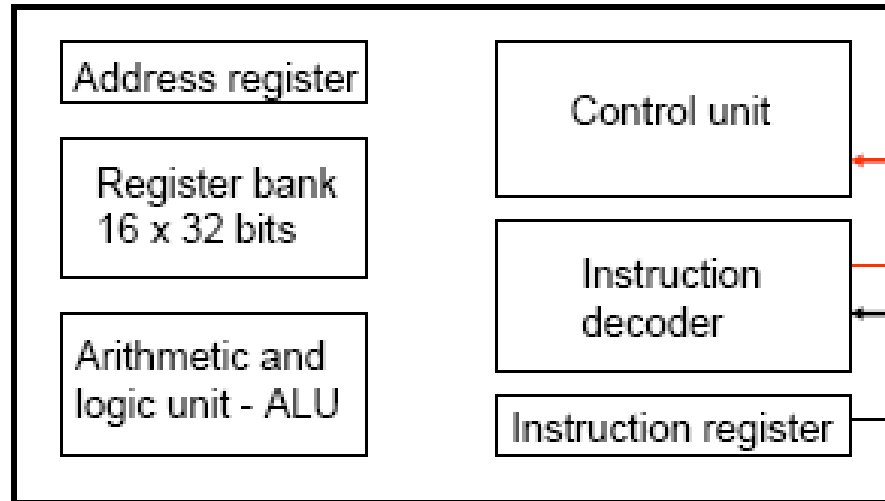
- The instruction decoder is connected to the control unit.

Internal connections

- The control unit is connected to the ALU, the register bank and the address register
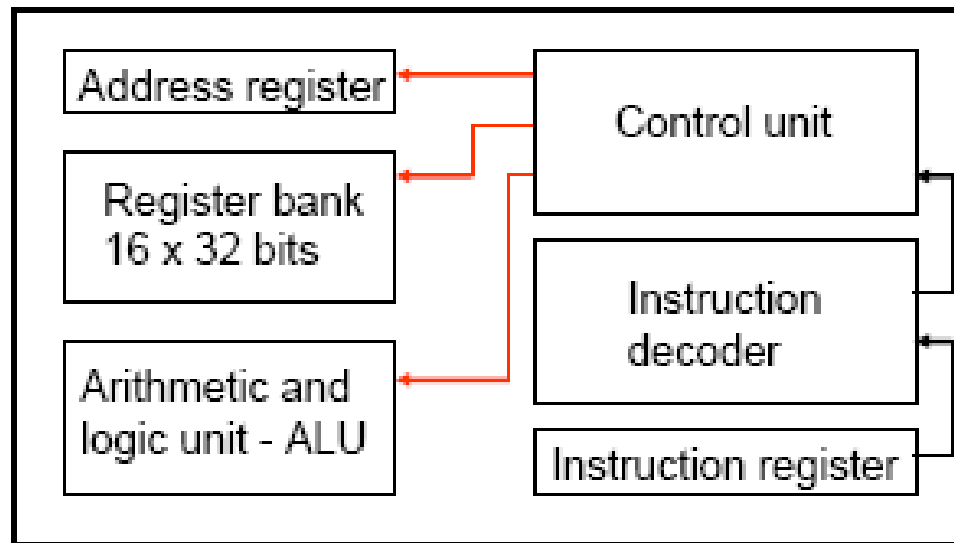
Internal connections

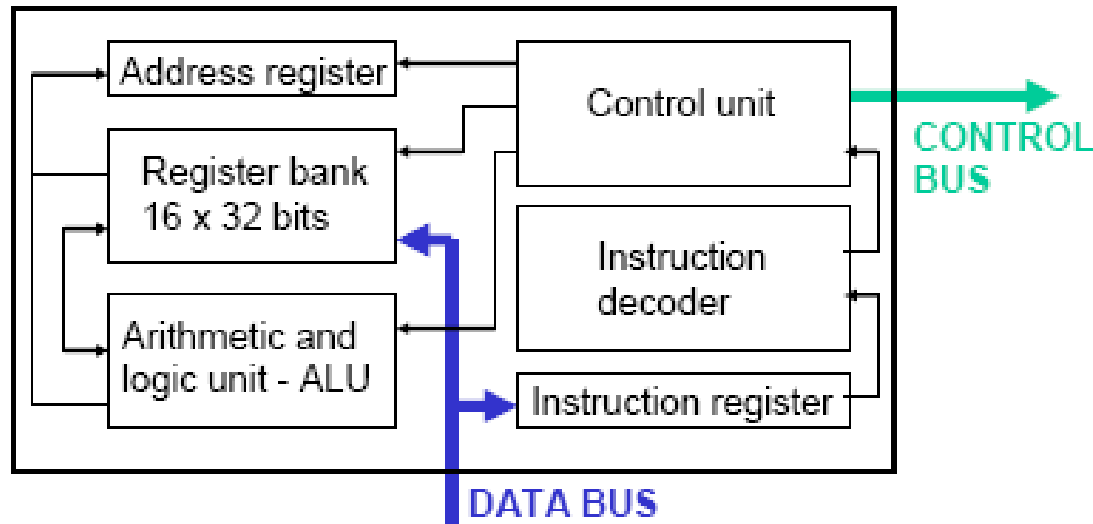- The ALU and the register bank are connected to each other and the address register.
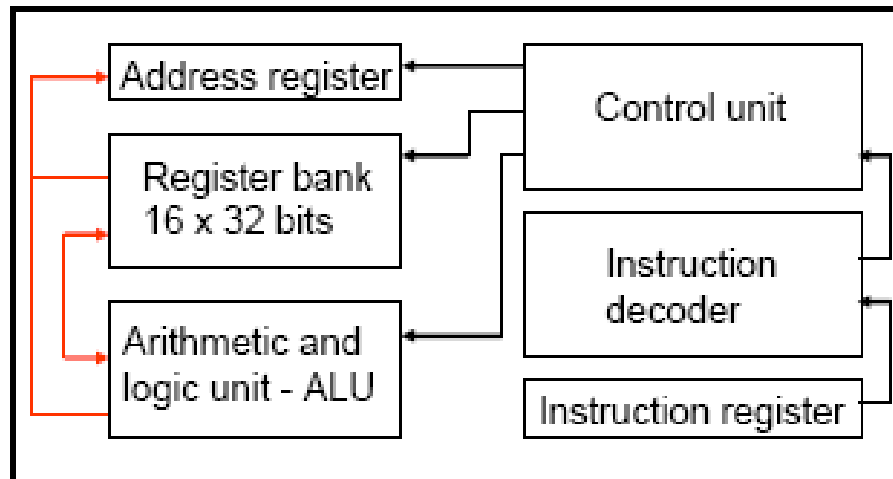
External connections

- The data bus is connected to both the instruction register and the register bank.
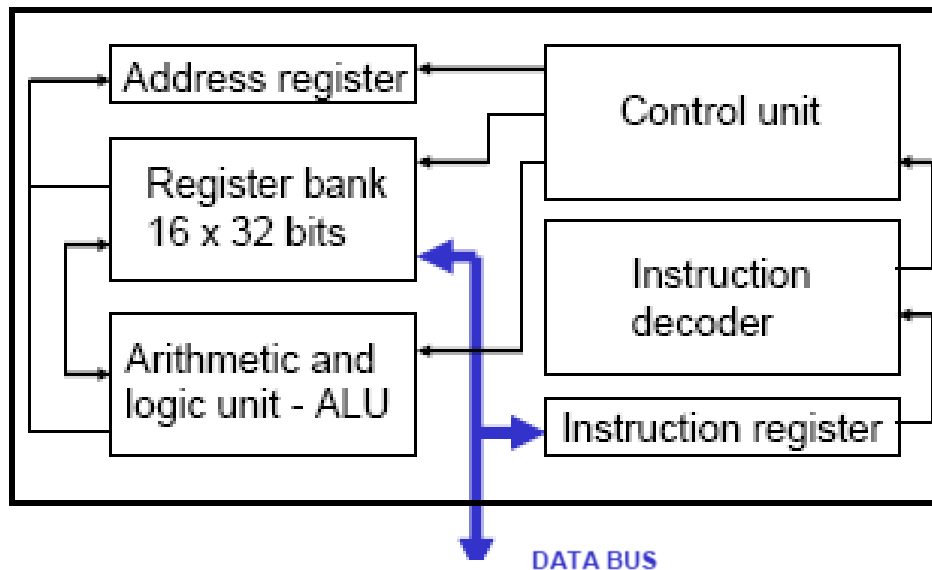
External connections

- The control bus is connected to the control unit.

External connections

- Address bus is connected to the address register
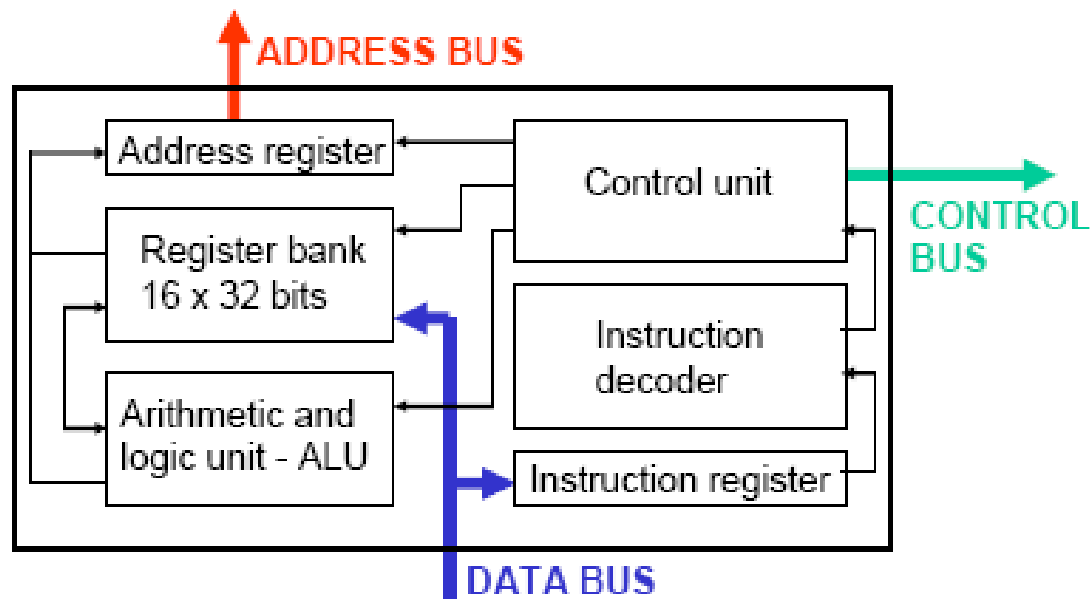
Program counter.

- Instructions are stored in memory so:
  - How does the CPU know the memory address for the next instruction in the computer program?
  - Register r15 always holds the memory address of the next instruction to be executed.
  - An alternative name for register r15 is the 'program counter'.

Instructions stored in memory.

- Instructions are 32 bits long, whereas memory locations are 8 bits long so one instruction occupies four locations in memory, e.g.
  - one instruction would be stored in 4 memory locations with addresses 0x00008000,0x00008001, 0x00008002 and 0x00008003.
  - The next instruction would be stored at addresses 0x00008004, 0x00008005, 0x00008006 and 0x00008007 and so on.

Instructions stored in memory.

- In general instructions are in consecutive locations in memory so that they are executed in the same order as they appear in memory.

  - If the instruction being executed is stored at address 0x00008000,

  - the next instruction to be executed will be stored at address 0x00008004,

  - the next at 0x00008008,

  - the next at 0x0000800C and so on – except when……

Program counter.

- When an instruction is executing, the program counter, r15, increments by 4 so that it holds the memory address of the next instruction.

- EXCEPT when a 'branch' instruction is executed when the computer program 'branches' to another part of memory and the program counter holds a completely new memory address.

Simple instructions.

- One of the simplest instructions is to move a value into a register e.g.
  - move 114 into register r12
  - The machine code for this instruction is 0xE3A0C072
  - After the instruction is executed
    - register r12 will hold the value 0x00000072 (hexadecimal for 114) and
    - value in register r15, the program counter, will have increased by 4.
    - All other registers remain unchanged.

Machine code.

- Look at the machine code for the instruction;
  - move 114 into register r12, again.
    - 0xE3A0C072
    - 1110 0011 1010 0000 1100 0000 0111 0010
  - The value 114 is given in the least significant byte
    - 0xE3A0C072
    - 114 in decimal is 72 in hexadecimal.
  - Register r12 is given by the 5th digit
    - 0xE3A0C072
    - 12 in decimal is C in hexadecimal.

Question

- Registers r7, r12 and r15 hold the values 0xCCDDEEFF, 0xFEDCBA98 and 0x00000108 respectively.

- What values are held by registers r7, r12 and r15 after the execution of the instruction with machine code 0xE3A070CB?

Another simple instruction.

- Another simple instruction is to move the value in one register to another register e.g.
  - move into r6 the value held in r14
  - The machine code for this instruction is
    - 0xE1A0600E
  - After the instruction is executed register
    - r6 will hold the same value as held in register r14 and
    - value in register r15, the program counter, will have increased by 4.
  - Other registers including r14 are unchanged.

Machine code.

- Look at the machine code for the instruction;
  - move into register r6 the value in register r14
    - 0xE1A0600E
  - Register r6 is given by the 5th hexadecimal digit
    - 0xE1A0600E
  - Register r14 is given by the last hexadecimal digit
    - 0xE1A0600E
    - 14 in decimal is E in hexadecimal.

Question

- Registers r7, r12 and r15 hold the values 0xCCDDEEFF, 0xFEDCBA98 and 0x0000010C respectively.

- What values are held by registers r7, r12 and r15 after the execution of the instruction with machine code 0xE1A0700C?