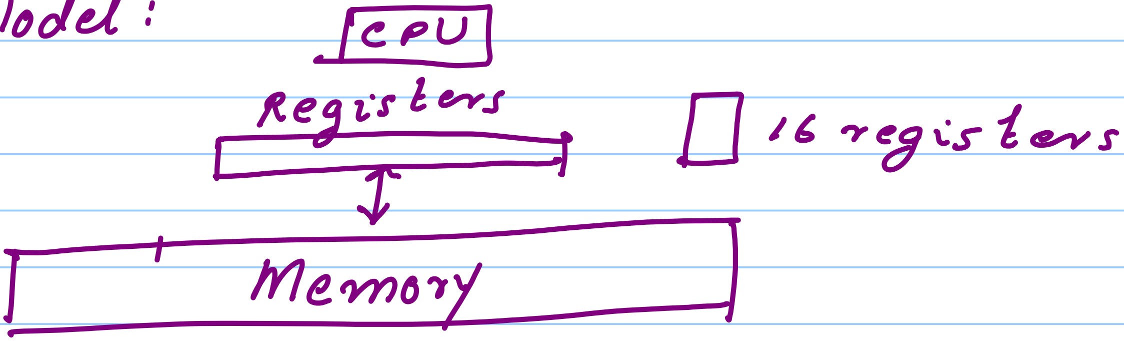
Aug 17th

Note Title 17-08-2011

1) Assembly Code Programming

Machine Model:



32 — 4GB
64 — 2^{64} byte mem. space

Register: Extremely fast storage media
1 register can contain upto 4 bytes of data

Administrative Trivia

1) Ensure that linux works on your laptop

2) Ubuntu → Dual boot
→ Vmplayer.

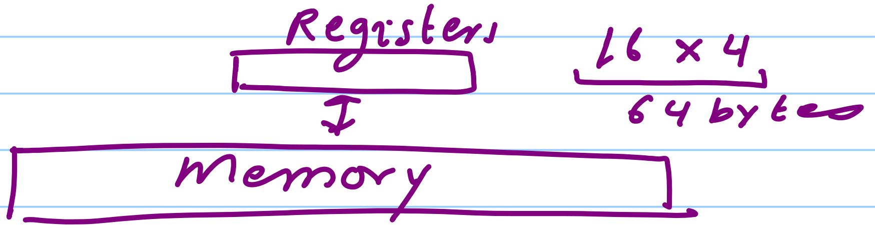
Thursday –	11-12 (busy)	
	3-4:30 (busy)	
Friday –	4:45 – 6:15 (busy)	

Do let me Know if you find a slot

1) Room is free
2) All of you are free

Mail me by today evening

Logical Organization of the memory system:



Very simple instruction:

C code.

$a = b + c;$

32 32 32

$\frac{96+k}{100+}$

Assembly Code

Register Mapping

$a \to r_1$, $b \to r_2$, $c \to r_3$

$\underbrace{\text{ADD } r_3, r_1, r_2}$

Concise representation of an instruction.

$$\left.\begin{array}{l} a \longleftrightarrow r_1 \longleftarrow 4 \\ b \longleftrightarrow r_2 \longleftarrow 4 \\ c \longleftrightarrow r_3 \longleftarrow 4 \end{array}\right\} 12 \text{ bits}$$

[Easily fit an instruction within 32 bits.

Conceptual Structure of the Program

1) Load data from memory into registers

2) Operate on registers

3) Store data from registers back to main memory.

Data Processing Instructions

ADD r_3, r_1, r_2

ADD → ARM assembly instruction

<instruction> <dest. reg.> <src. reg.$_1$> <src. reg$_2$>

r_3 → destination register

r_2 → source reg 1

r_1 → source reg 2

Other DP Instructions

SUB r_3, r_1, r_2 $(r_3 = r_1 - r_2)$

MUL r_3, r_1, r_2 $(r_3 = r_1 \times r_2)$

Divide → ARM instruction set before version 7 did not have divide

→ After version 7

udiv r_3, r_1, r_2 $(r_3 = r_1 / r_2)$

(Logical Shift Left)

Assembly $\underbrace{\text{LSL } r_3, r_1, r_2}$ $\underbrace{(r_3 = \overset{\swarrow 2}{r_1} << \overset{\swarrow 3}{r_2})}$

(Logical Shift Left)

$r_3 = 2 \times 8 = 16$

LSR r_3, r_1, r_2 $(r_3 = r_1 >> r_2)$

Right Shift ASR r_3, r_1, r_2 $(r_3 = r_1 >> r_2)$
(−2)

(w/. sign extension)

1110 (−2)
$\rightarrow$

1 ~~0~~111 (+3)

Sign Extension: Right Shift a (+)ve number
add 0 in the MSB

a (−) ve number
add 1 in the MSB

Arithmetic Shift Right

LSR $\rightarrow$ assumes that num. is unsigned.

ASR $\rightarrow$ number is signed

Summary: LSL, LSR, ASR (Shift)

ADD, SUB, MUL, UDIV, SDIV (Arithmetic)

↗ unsigned ↑ signed

Logical Instructions:

ORR r_3, r_1, r_2 ($r_3 = r_1 \mid r_2$)

AND r_3, r_1, r_2 ($r_3 = r_1 \& r_2$)

Format: Both operands were registers

$c = a + 2$

ADD r_3, r_1, #2 ($r_3 = r_1 + 2$)

2 → constant
(immediate value)

🚩 ADD r_3, #4, #5 ✗ (Not allowed)

The second operand can be an immediate (number) in all the instructions that we have studied uptil now.

LSL r_3, r_1, #2

$r_3 = r_1 << 2$

$r_3 = r_1 + r_2$
(register format)

$r_3 = r_1 + 2$
(immediate form

Extended / shifted Format

ADD r_3, r_1, r_2, $\begin{bmatrix} LSL \\ LSR \\ ASR \\ ROR \end{bmatrix}$ #$\textcircled{2}$ $\xleftarrow{}$ (< 32)

$$r_3 = r_1 + \begin{cases} r_2 << 2 & \text{— LSL} \\ r_2 >> 2 & \text{— LSR} \\ r_2 >> 2 & \text{— ASR (Sign Extn)} \\ r_2 \text{ ROR } 2 & \text{— ROR} \end{cases}$$

MOV & MVN instruction

R : MOV r_3, r_2 $(r_3 = r_2)$

I : MOV r_3, #5 $(r_3 = 5)$

S: mov r_3, r_1, LSL #2 $(r_3 = r_1 << 2)$

MVN : Move not   Not → One's Comp.

Not 000 → 111

R: MVN r_3, r_2 $(r_3 = \sim r_2)$ Not 010 → 101

MVN r_3, #1 $(r_3 = \sim 1)$

MVN r_3, r_2, LSL #2 $(r_3 = \sim (r_2 << 2))$