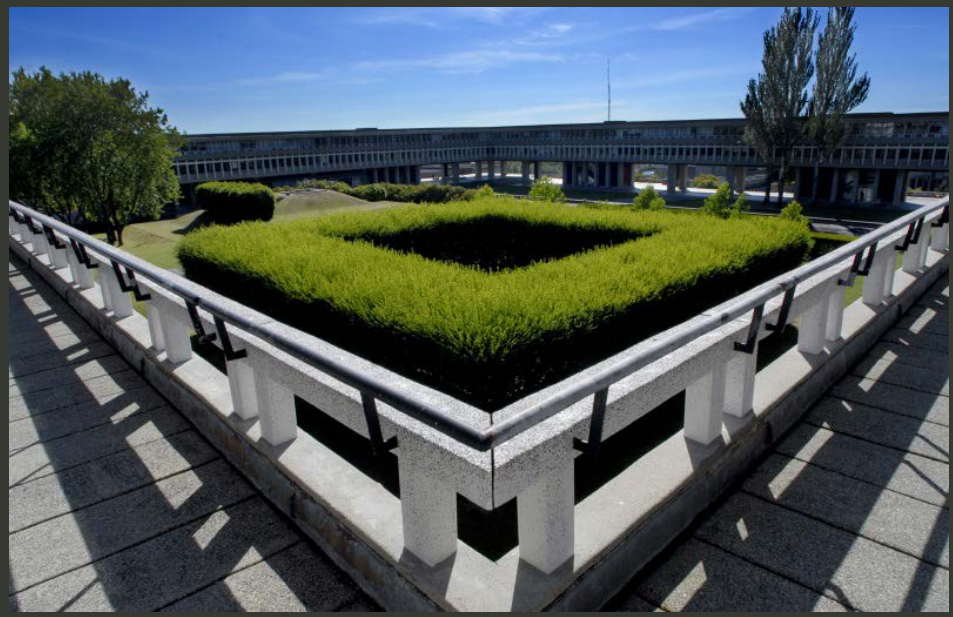
SFU SIMON FRASER UNIVERSITY
ENGAGING THE WORLD

ENSC254 – Assembly Language and Programming Modes

© 2017 -- Fabio Campi and Craig Scratchley
School of Engineering Science
Simon Fraser University
Burnaby, BC, Canada



ARM Processor Modes

- We will focus now on the ARM ISA Version 4. Newer versions of the ISA may have at least a few differences.

  - ***PROCESSOR MODE is stored in the CPSR status register of the processor, and it impacts the way the processor reacts to given inputs***
  - Much like a person's Mood: if a student is in a good mood, he/she will react well to this lecture
  - Much like a person's mood, a PROCESSOR'S MODE can be changed by persuasion (software, in the case of a processor), but is more likely to change due to external events
  - ARM ISA v4 PROCESSOR MODES:
    1) SVC (Supervisor)
    2) FIQ (High Priority (FAST) Interrupt Raised)
    3) IRQ (Low priority Interrupt Raised)
    4) Abort (Memory Access violation)
    5) Undef (Undefined Instruction)
    6) System
    7) User (could be considered mode 6b)

Modes 1 to 6(a) are defined to be privileged modes

Note: You will need to understand the above, BUT you do not need to memorize them

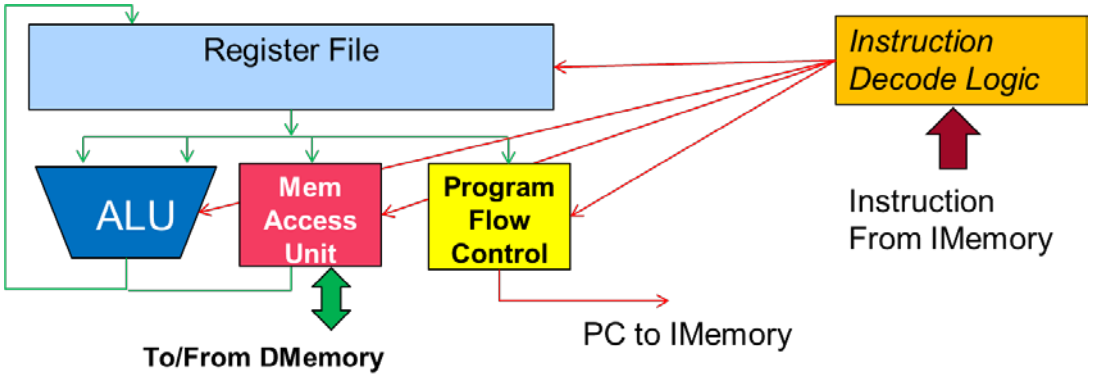
Do you know what an INTERRUPT is

A. I pour interrupts on my pancakes every morning!
B. Yes, but I wouldn't mind some review
C. I have a vague notion
D. Never heard of them

ARM Registers

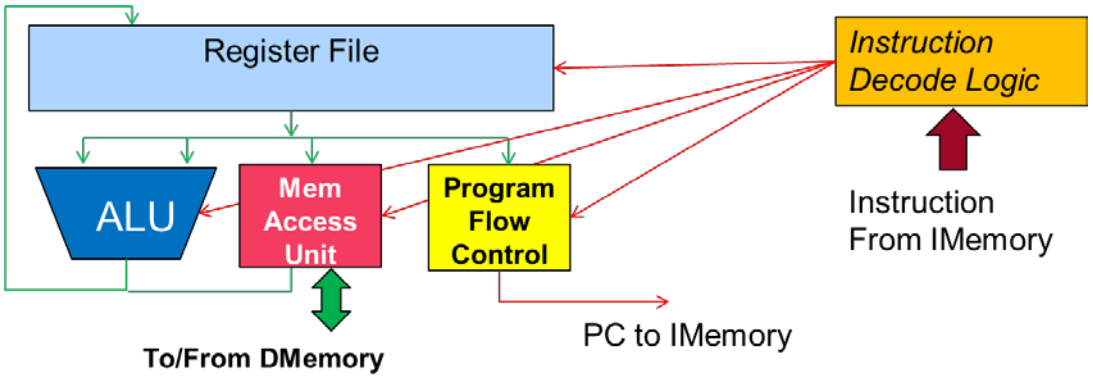
Registers are the ***PRIMARY STORAGE AREA*** for the processor

- In ARM7TDMI there are actually 37 Registers
  - The Program Counter (PC)
  - 30 other “General Purpose” Registers (GPRs)!
  - 6 Status Registers (SR)!



ARM GPR Registers

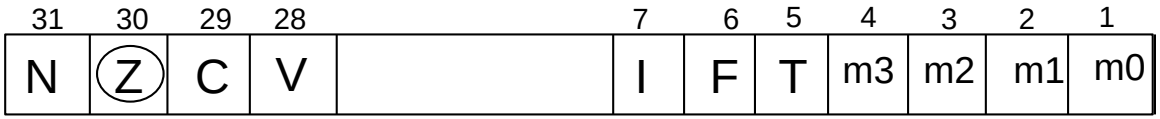
- GPRs are 32-bit, and are addressed in Assembly with the keywords r0, r1,..
- Access to registers is different depending on the processor MODE!
  - ***At any one time, 16 GPRs (r0,…,r15) plus 1 or 2 SRs are visible and accessible***
  - Some of the GP registers have a typical or specific function:
    - R13 ➔ typically for Stack Pointer
    - R14 = Link Register
    - R15 = Program Counter



ARM Program Status Register

- The ARM processor has specific features designed to optimize code size and power consumption, that are critical parameters for embedded systems
- In particular, the ARM register file features a specific STATUS REGISTER that centralizes all information regarding the state and the “programming mode” of the processor
- In most other RISC processors, the state is saved as information in general purpose registers

ARM Program Status Register(1)



NZCV are ***FLAG*** bits: Flags in CPSR are changed by ALU operations, and they may be read as a condition for other operations (i.e. to implement an if statement)

This is a commonly used feature of the ARM processor, it is a very Un-RISC feature but it is used to allow for very compact code:

C Code

```
main() {
  if (a>b) c=a-b;
  else c=b-a;
}
```

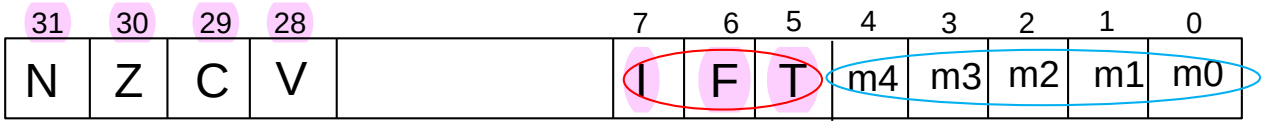
MIPS Asm Code

```
  sgt r2,r3,r4
  bnz r2,label1
  sub r0,r4,r3
Label1:
  bz r2,label2
  sub r0,r3,r4
Label2:
  [..]
// sgt is “set greater than”
```

ARM Asm Code

```
subs r0,r3,r4
sublt r0,r4,r3
[..]
```

ARM Program Status Register(2)



I|F|T are ***STATUS*** bits:

I and F are changeable only in privileged mode, set by specific write operations and determine the behavior of the processor:

- I = Interrupt mask (Disable normal Interrupts)
- F = Fast Interrupt Mask (Disable Fast Interrupts)
- T = Thumb, activates compact 16-bit Thumb instruction set

M[4…0] represents the *Processor Mode*, according to the following translation

10000 User Mode
10001 FIQ Mode
10010 IRQ Mode
10011 Supervisor Mode
10111 Abort Mode
11011 Undefined Mode
11111 System Mode

The Vector Table

What happens when we switch on a computer? What will the processor do?

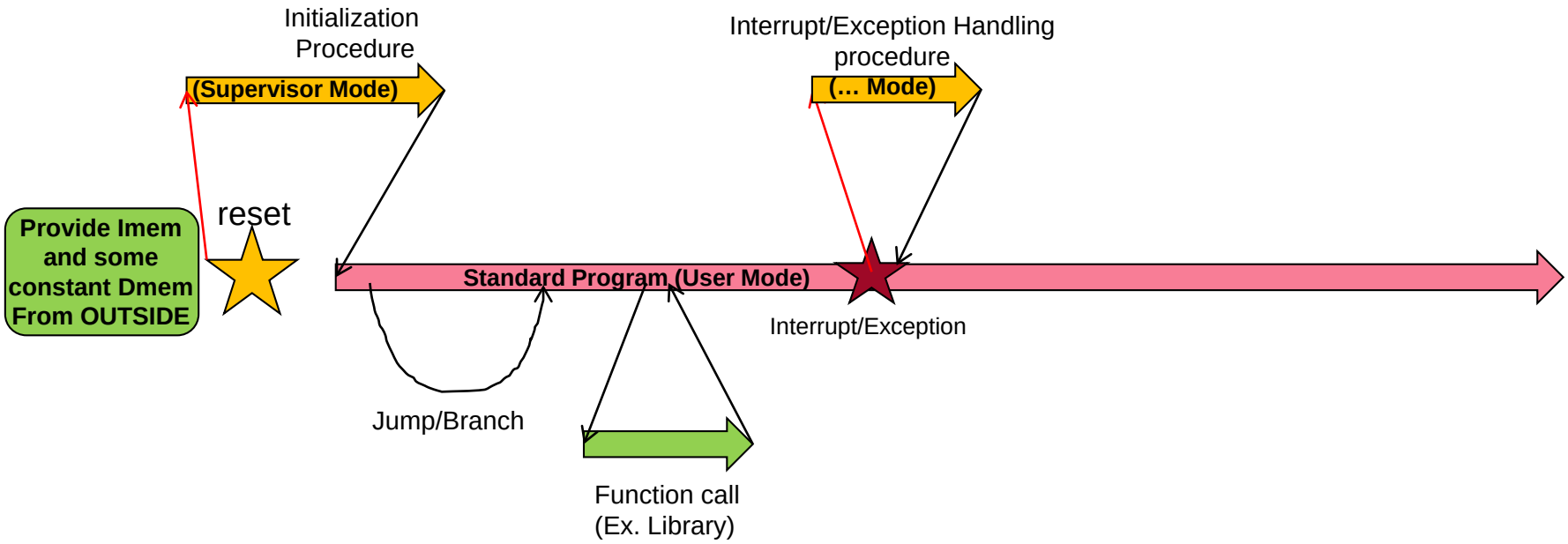
THINK HARDWARE! To power up means communicating a reference Voltage to all circuits in the system. If we don't control them, the circuits will just get random states.

The Program Counter will hold a randomized value and the processor will start executing at that random location in the memory.

- WE NEED TO RESET the system. That is, upon power-up, set its status to a known condition
- Similarly, when we INTERRUPT the processor's functioning, we want to direct it to a routine starting at a specific location in the address space.

In the case of any unexpected or "Asynchronous" event, the first thing to do is set the Program Counter (PC) one of a small set of known locations so that the processor can (start or) continue its execution flow from there

Processor Computation Flow



Calls to subroutines, and all *returns* are always handled by specific instructions that are part of the code and depicted as black arrows. Same for Software Interrupt.

In case of asynchronous events (Reset, IRQ, FIQ) we must define a way to set the program counter with the desired address (Red arrows)

The Vector Table (2)

- We call the VECTOR TABLE, or interrupt table, a set of specific "recovery addresses" for each asynchronous (or SWI) event handled by a processor. ***Upon the issue of a reset, interrupt, or exception a processor will typically branch to the specific location contained in the TABLE and start executing from there***

Reset	0x000000	***Reset***
Undefined Instruction	0x000004	Undef
Software Interrupt (SWI)	0x000008	SVC
Instr Memory Abort	0x00000C	Abort
Data Memory Abort	0x000010	Abort
IRQ	0x000018	IRQ
FIQ	0x00001C	FIQ

- In the cases of Reset/Hardware-Interrupt/Exception, the processor mode is set to the appropriate mode, and a processor's execution will typically jump to the address specified above.
- A specific feature of ARM is that the memory locations in the TABLE (0x0 to 0x1C) actually contain instructions, often BRANCH instructions to the handling code.

Most processors instead contain addresses. This potentially allows for faster interrupt handling: rather than loading the address and jumping there, ARM simply executes the instruction.