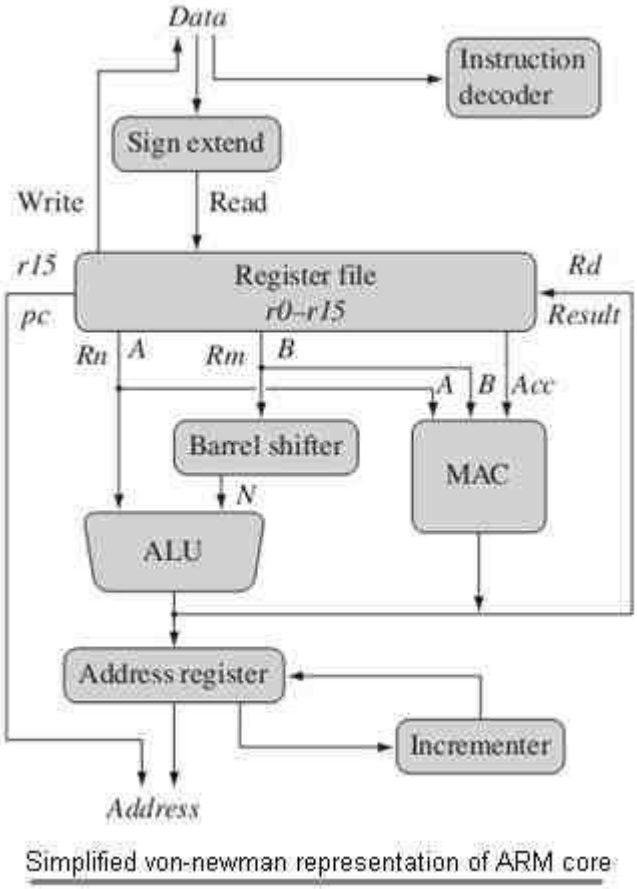
TI DSP, MCU, Xilinx Zynq FPGA Based Programming Expert Program

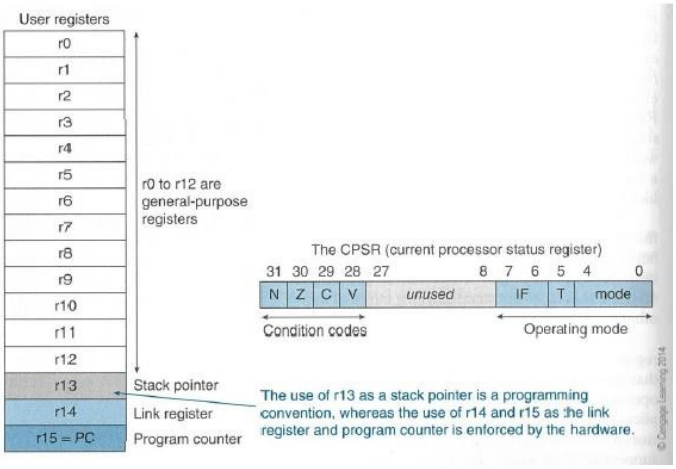
Instructor – Innova Lee (Sanghoon Lee)
gcccompil3r@gmail.com
Student – Howard Kim (Hyungjoo Kim)
mihaelkel@naver.com

ARM Processor

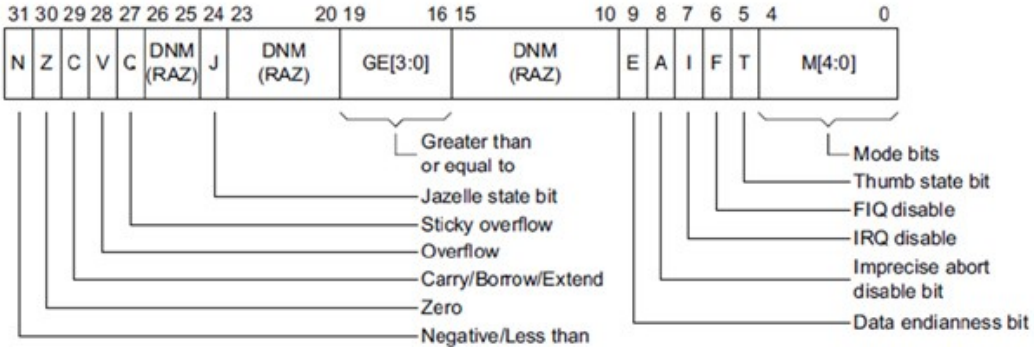


Simplified von-newman representation of ARM core

Above is a map for ARM core.
There are 16 pieces of general purpose registers, r0-r15.
Barrel shifter is for shift operation.
ALU is for calculation.
Data bus on the top is for input, Address bus is for output.
MAC which is optional can process multiple operation, this enable the process be much more faster.
When coding with Assembly Language, We use general purpose registers. Let’s find out!



r0-r12 is general purpose registers, r13 is Stack pointer, r14 is Link register, r15 is Program counter. Else, there are 2 status register, cpsr(current program status register) and spsr(saved program status register).



N (Negative) : set if result from an operation is negative(-)
Z (Zero) : set if result from an operation is 0
C (Carry) : set if an operation causes Carry
V(oVerflow) : set if an operation causes signed overflow
Q(Saturation) : set if an operation causes overflow or saturation.

<OpCode>	Instruction	Description
0000	AND	Rd:=Op1 AND Op2
0001	EOR	Rd:=Op1 XOR Op2
0010	SUB	Rd:=Op1 - Op2
0011	RSB	Rd:=Op2 - Op1
0100	ADD	Rd:=Op1 + Op2
0101	ADC	Rd:=Op1 + Op2 + C
0110	SBC	Rd:=Op1 - Op2 + C - 1
0111	RSC	Rd:=Op2 - Op1 + C - 1
1000	TST	Op1 AND Op2 -> CPSR
1001	TEQ	Op1 XOR Op2 -> CPSR
1010	CMP	Op1 - Op2 -> CPSR
1011	CMN	Op1 + Op2 -> CPSR
1100	ORR	Rd:=Op1 OR Op2
1101	MOV	Rd:=Op2
1110	BIC	Rd:=Op1 AND (NOT Op2)
1111	MVN	Rd:=NOT Op2

Above is some instructions.

Implement multiple-table with asm.

test.c (~/HomeworkBackup/44th) - VIM

```c
#include <stdio.h>


int main(void){

    register unsigned int r0 asm("r0") = 0;
    register unsigned int r1 asm("r1") = 0;
    register unsigned int r2 asm("r2") = 0;
    register unsigned int r3 asm("r3") = 0;
    register unsigned int r6 asm("r6") = 0;
    register unsigned int r7 asm("r7") = 0;
    register unsigned int r8 asm("r8") = 0;

    printf("9단! :\n");
    asm volatile("mov r6,#9");
    asm volatile("mov r7,#1");
    asm volatile("mov r8,#0");

    asm volatile("R1:");
    asm volatile("add r8,r8,r6");
    printf("%d*%d=%d\n",r6,r7,r8);

    asm volatile("add r7,r7,#1");
    asm volatile("cmp r7,#9");
    asm volatile("ble R1");


    return 0;
}
```

Result

```
howi@ubuntu:~/HomeworkBackup/44th$ arm-linux-gnueabi-gcc -g test.c
howi@ubuntu:~/HomeworkBackup/44th$ qemu-arm-static -L /usr/arm-linux-gnueabi ./a.out
9단! :
9*1=9
9*2=18
9*3=27
9*4=36
9*5=45
9*6=54
9*7=63
9*8=72
9*9=81
```