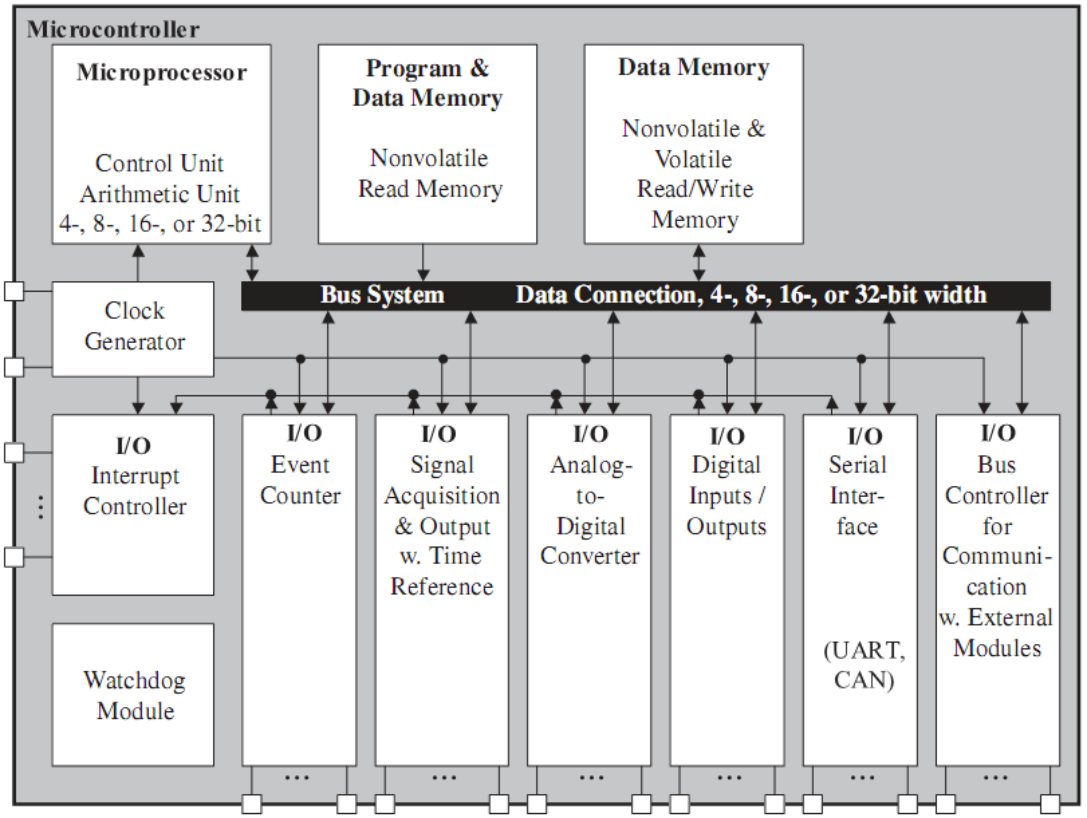
소프트웨어 개발자를 위한 임베디드 시스템 하드웨어

Embedded System

	PC	GPU(TPU/NPU)	AP	High-end MCU	MCU	Automotive MCU	Remark
CPU	Intel CORE i9	Tensor Processor Neural Processing Unit	Cortex A11	Cortex-M7 Cortex-R	Cortex-M3/M0		
# of CPU	16	128	4	1	1	2 ~ 3	Dual Core Lock Step
Frequency	5GHz	2GHz	2GHz	400MHz	8MHz ~ 200MHz		
OS	Windows 11	-	Andriod	Linux	Real Time OS	AutoSAR	
Program Memory	L1, L2 Cache Mbytes	L1, L2 Cache Mbytes AI Cache	L1, L2 Cache Mbytes	Cache Kbytes	1Kbytes ~ 1Mbytes (Embedded FLASH)	16Kbyts ~ 4Mbytes (Embedded FLASH, ECC)	OTA(Over The Air)
Data Memory					256Bytes ~ 1Mbytes Cache SRAM Scratch SRAM	ECC Protected Kbytes ~ 1Mbytes Cache SRAM Scratch SRAM	
DRAM	64Gbytes	16Gbytes	16Gbytes	4Gbytes	-	-	
HDD	NAND	NAND	NAND	NAND	Serial NAND FLASH Serial NOR FLASH		
Power	300W	60W	30W	10W	1mW ~ 2W		
PCB	20cm x 20cm	20cm x 10cm	10cm x 10cm	5cm x 5cm	2cm x 2cm ~ 5cm x 5cm		
Technology	7nm Digital	7nm Digital	14nm Digital	55nm Analog/Digital	55nm ~ 180nm Analog/Digital/Power		

마이크로컨트롤러(MCU)



C → Assembly → Link → .hex

Cpu0_Main.c | Cpu2_Main.c | IfxCpu_CStart0.c | IfxScuWdt.h | string.h | IfxCpu_Trap.c | IfxCpu.h

```
51      * Enable the watchdogs and service them periodically if it is required
52      */
53     IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword());
54     IfxScuWdt_disableSafetyWatchdog(IfxScuWdt_getSafetyWatchdogPassword());
55
56     /* Wait for CPU sync event */
57     IfxCpu_emitEvent(&g_cpuSyncEvent);
58     IfxCpu_waitEvent(&g_cpuSyncEvent, 1);
59
60     printf("Hello World\n");
61
62     for( int i=0; i<0x2000; i++)
63     {
64         *((volatile int *)0x70008000+i) = 0x70800000 + i;
65         *((volatile int *)0x60008000+i) = 0x60800000 + i;
66     }
67
68     // CPU0 Data Scratch-Pad RAM
69     systemtick[0] = SYSTEM_TIMER_31_0;
70     checksum_0 = 0;
71     for( int i=0; i<0x2000; i++)
72         checksum_0 += *((volatile int *)0x70008000+i);
73     systemtick[1] = SYSTEM_TIMER_31_0;
74
75     // CPU1 Data Scratch-Pad RAM
76     systemtick[2] = SYSTEM_TIMER_31_0;
77     checksum_1 = 0;
78     for( int i=0; i<0x2000; i++)
79         checksum_1 += *((volatile int *)0x60008000+i);
80     systemtick[3] = SYSTEM_TIMER_31_0;
81
82     printf("0x7000 access @ cpu0 : %d\n", systemtick[1]-systemtick[0]);
83     printf("0x6000 access @ cpu0 : %d\n", systemtick[3]-systemtick[2]);
84
85     systemtick[0] = SYSTEM_TIMER_31_0;
86     memcpy((char *)0x70008000,(char *)0x70008000, 0x8000);
87     systemtick[1] = SYSTEM_TIMER_31_0;
88
89     systemtick[2] = SYSTEM_TIMER_31_0;
90     memcpy((char *)0x60008000,(char *)0x60008000, 0x8000);
91     systemtick[3] = SYSTEM_TIMER_31_0;
```

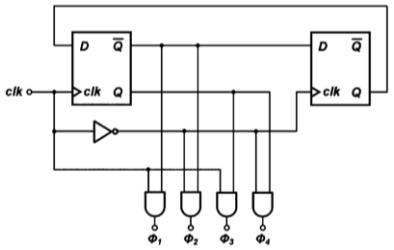
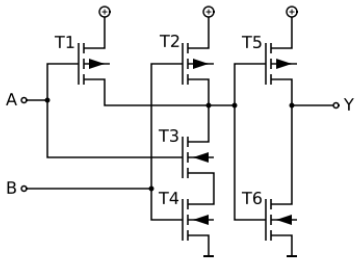
Disassembly | Outline — Enter location here

```
 65                 *((volatile int *)0x60008000+i) = 0x60800000 + i;
000000008000024e:   mul         d0,d15,#0x4
0000000080000252:   mov.a       a15,d0
0000000080000254:   movh.a      a3,#0x6001
0000000080000258:   add.a       a3,a15
000000008000025a:   lea         a15,[a3]-0x8000
000000008000025e:   movh        d0,#0x6080
0000000080000262:   add         d0,d15
0000000080000264:   st.w        [a15],d0
 62               for( int i=0; i<0x2000; i++)
0000000080000266:   add         d15,#0x1
0000000080000268:   mov         d0,#0x2000
000000008000026c:   jlt         d15,d0,0x80000236
 69               systemtick[0] = SYSTEM_TIMER_31_0;
0000000080000270:   movh.a      a15,#0x6000
0000000080000274:   lea         a15,[a15]0xe4
0000000080000278:   ld.w        d15,0xf0000010
000000008000027c:   st.w        [a15],d15
 70               checksum_0 = 0;
000000008000027e:   movh.a      a15,#0x6000
0000000080000282:   lea         a15,[a15]0xdc
0000000080000286:   mov         d15,#0x0
0000000080000288:   st.w        [a15],d15
 71               for( int i=0; i<0x2000; i++)
000000008000028a:   mov         d0,#0x0
000000008000028c:   jg          0x800002b8
 72                   checksum_0 += *((volatile int *)0x70008000+i);
000000008000028e:   movh.a      a15,#0x6000
0000000080000292:   lea         a15,[a15]0xdc
0000000080000296:   movh.a      a2,#0x6000
000000008000029a:   lea         a2,[a2]0xdc
000000008000029e:   ld.w        d15,[a2]
00000000800002a0:   mul         d1,d0,#0x4
00000000800002a4:   mov.a       a2,d1
00000000800002a6:   movh.a      a3,#0x7001
00000000800002aa:   add.a       a3,a2
00000000800002ac:   lea         a2,[a3]-0x8000
00000000800002b0:   ld.w        d1,[a2]
00000000800002b2:   add         d15,d1
00000000800002b4:   st.w        [a15],d15
 71               for( int i=0; i<0x2000; i++)
```

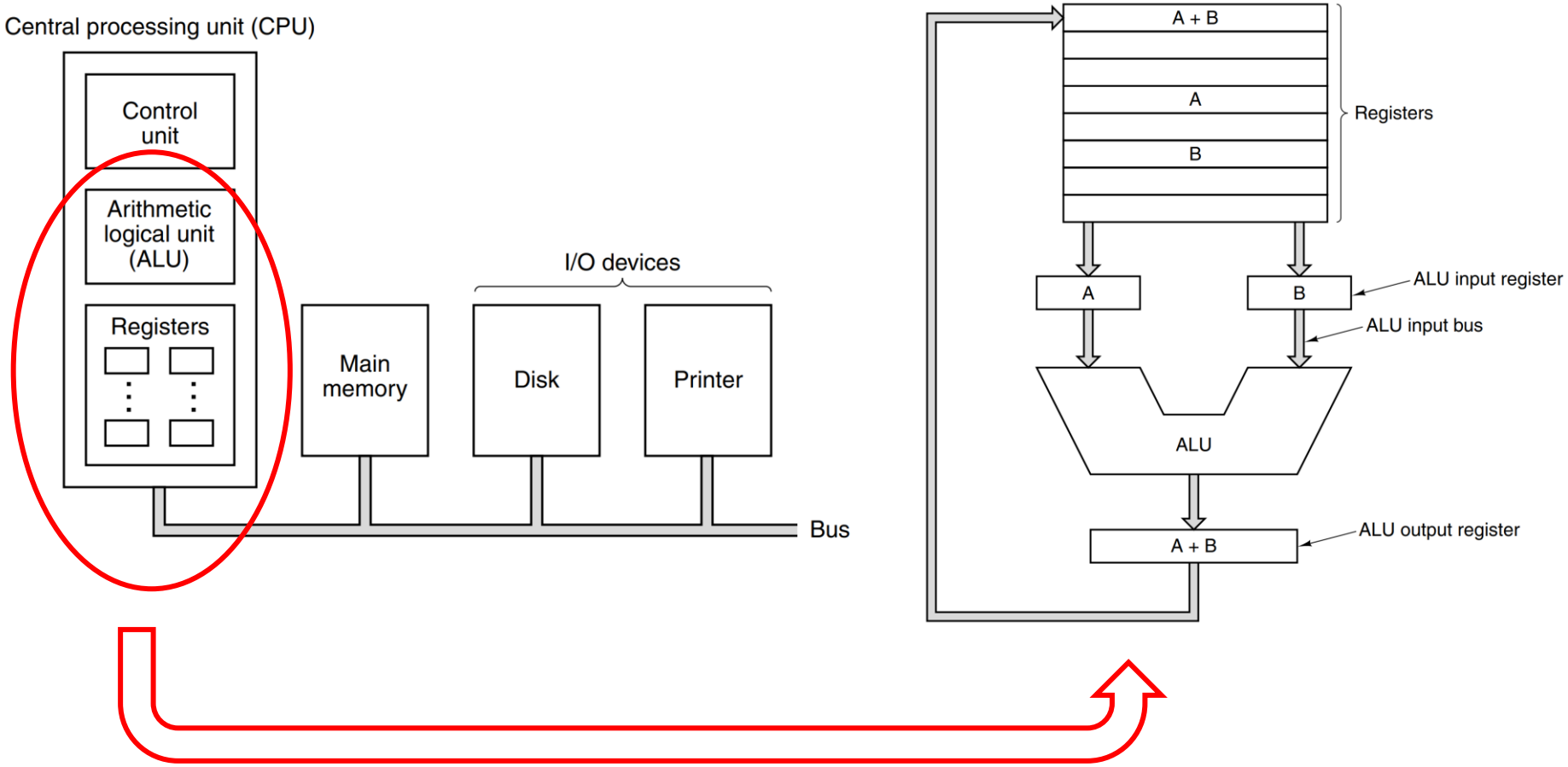
ACE Lab.

KNU

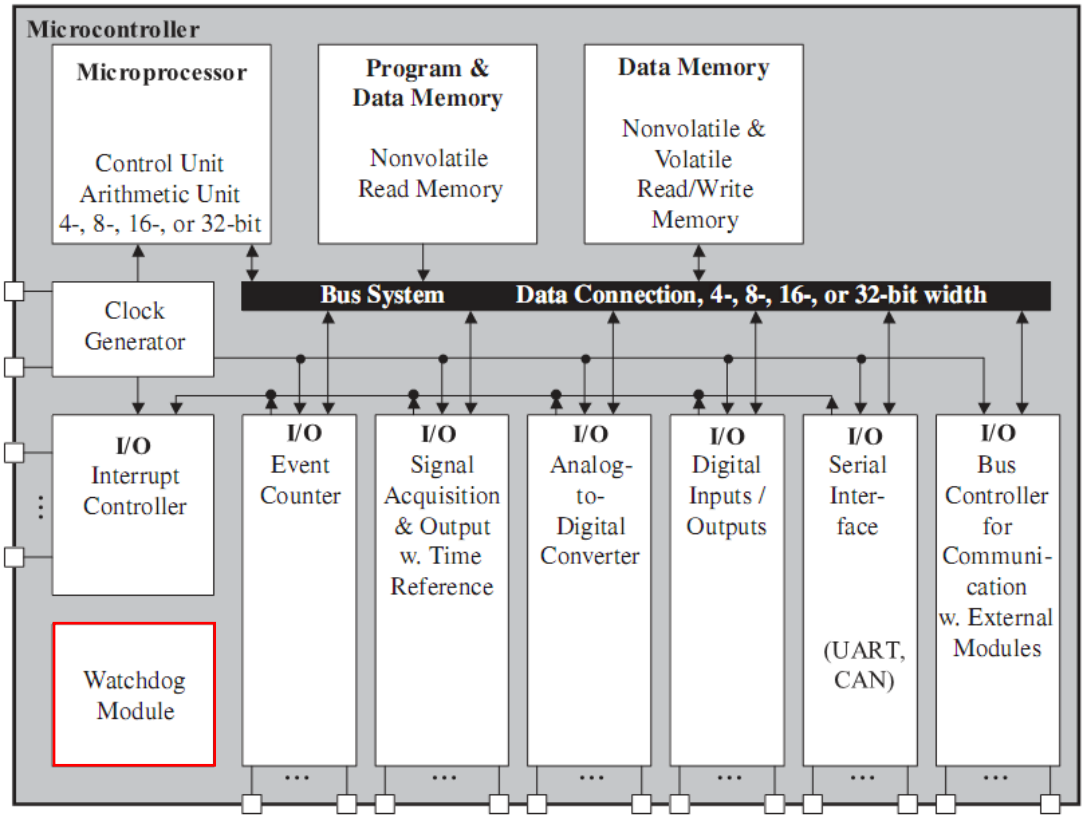
HW & SW 추상화 단계

High Level	sum = sum + 1;
Assembly	add r2, r2, 1
Machine	1101 0001 0011 0010 0010 0000 0000 0000 0001
Register Transfer	Fetch Instruction, Increment PC, Load ALU with r2 …
Gate	
Circuit	

CPU (Central Processing Unit)



마이크로컨트롤러(MCU)



Timer

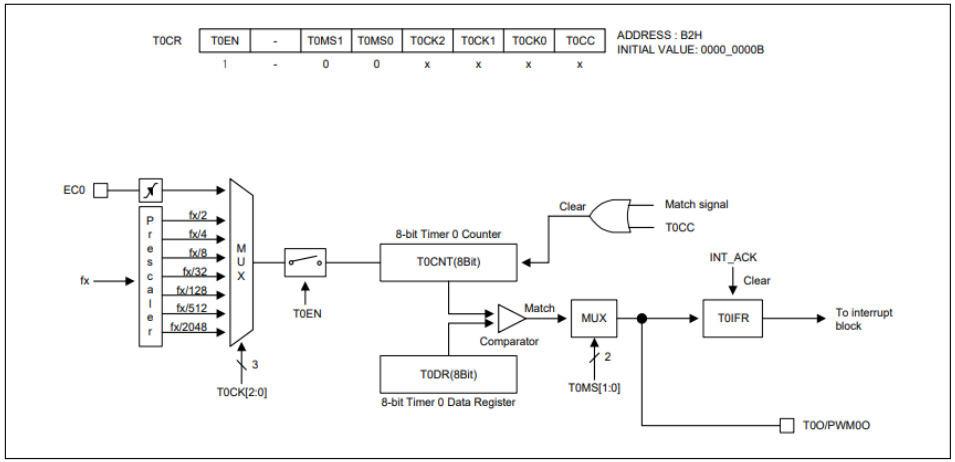


Figure 11.6 8-bit Timer/Counter Mode for Timer 0

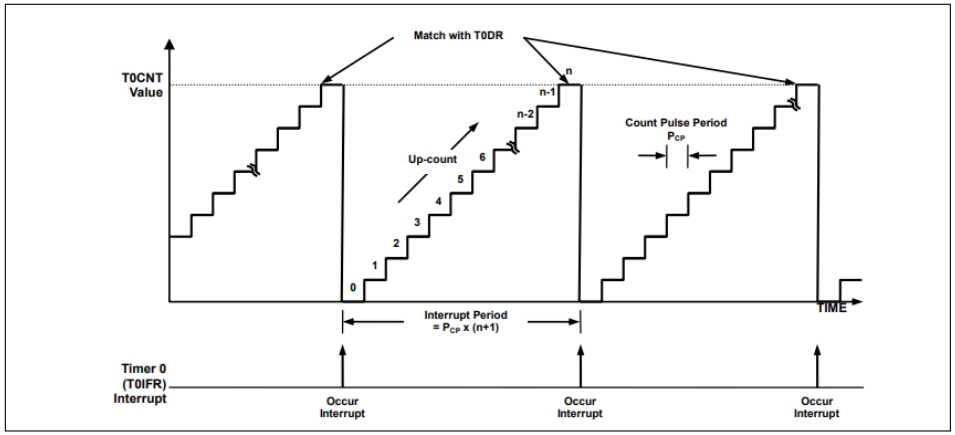


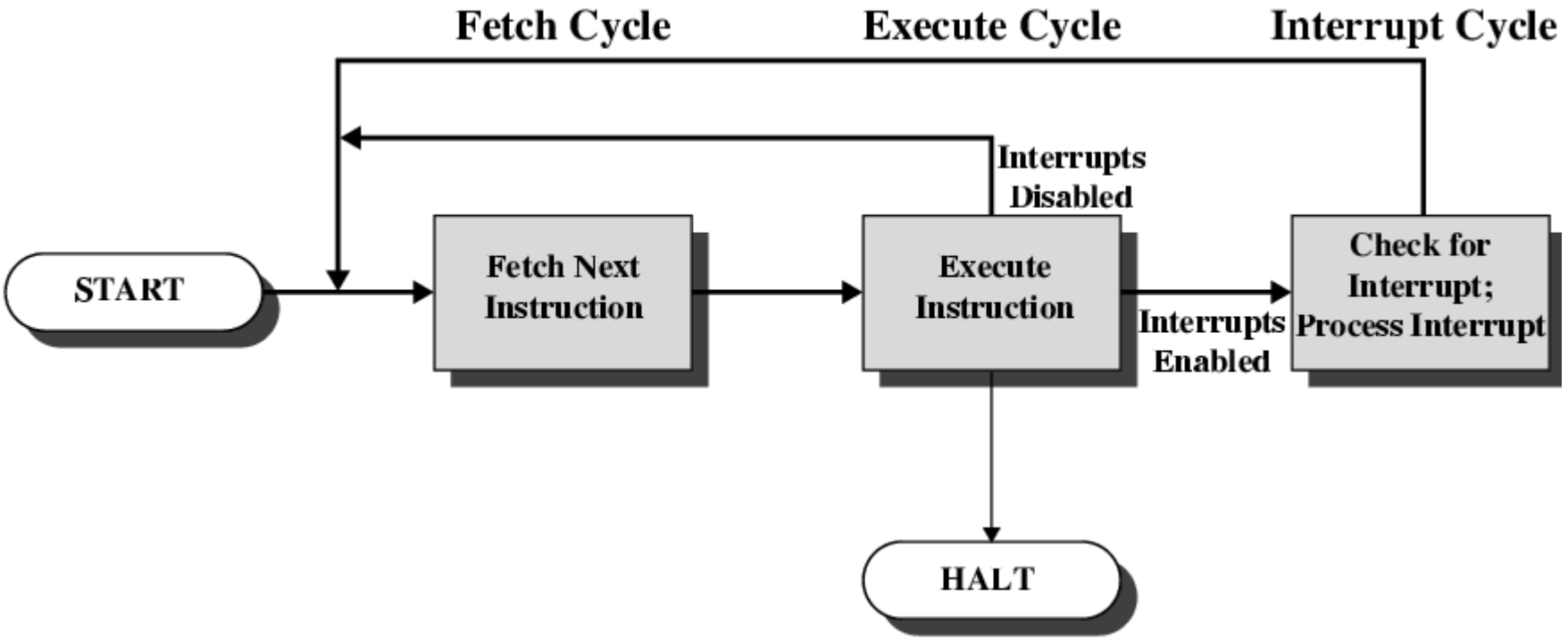
Figure 11.7 8-bit Timer/Counter 0 Example

System Watchdog vs. Safety Watchdog

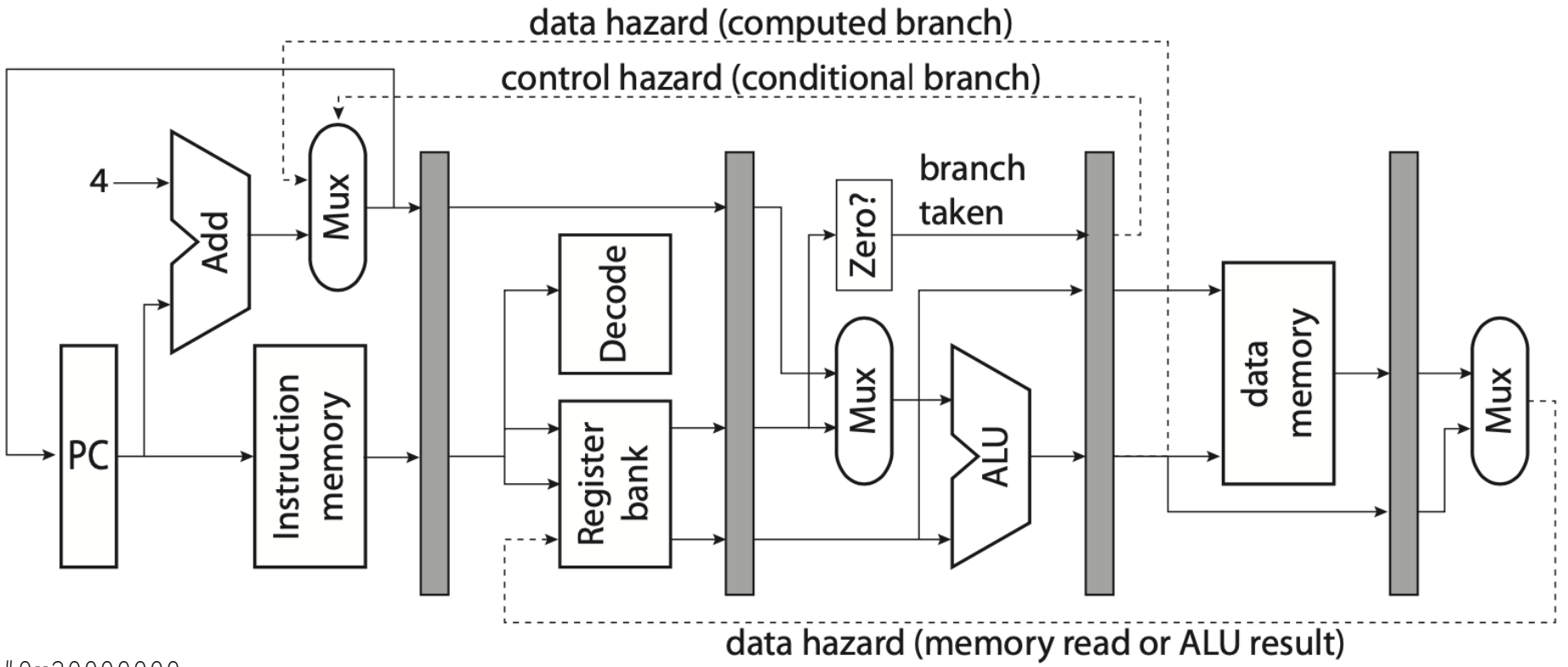
- System Watchdog
  - 특정 시간 동안 RELOAD 하지 않으면 System RESET 발생
  - System Mal-Function ➔ Auto Reset

- Safety Watchdog
  - 특정 시간 동안 RELOAD 하지 않으면 Safety LOCK 발생
  - System Safety, Block Unwanted System Register Write

명령어 실행

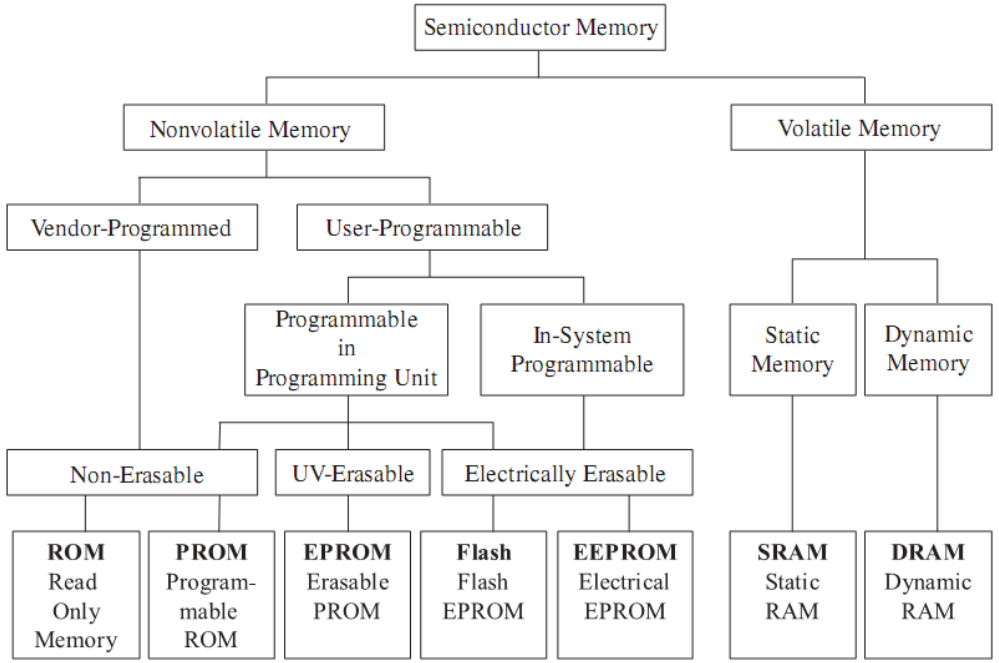

Fetch Cycle
Execute Cycle
Interrupt Cycle
Interrupts Disabled
START
Fetch Next Instruction
Execute Instruction
Interrupts Enabled
Check for Interrupt; Process Interrupt
HALT

파이프라인



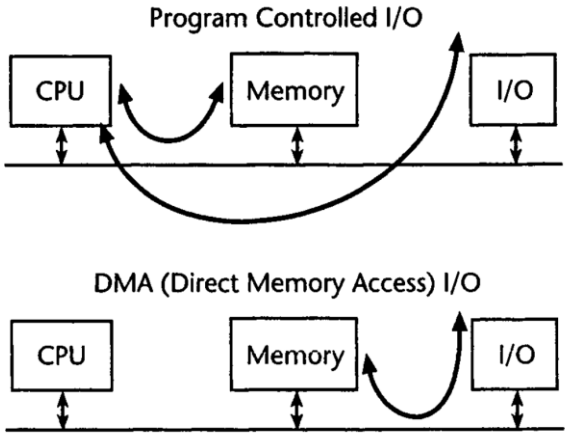

```
LD R1,#0x20000000
LD R2,#0x00010000
LD R0,R1
OR R0,R0,R2
ST R0,R1
```

반도체 메모리

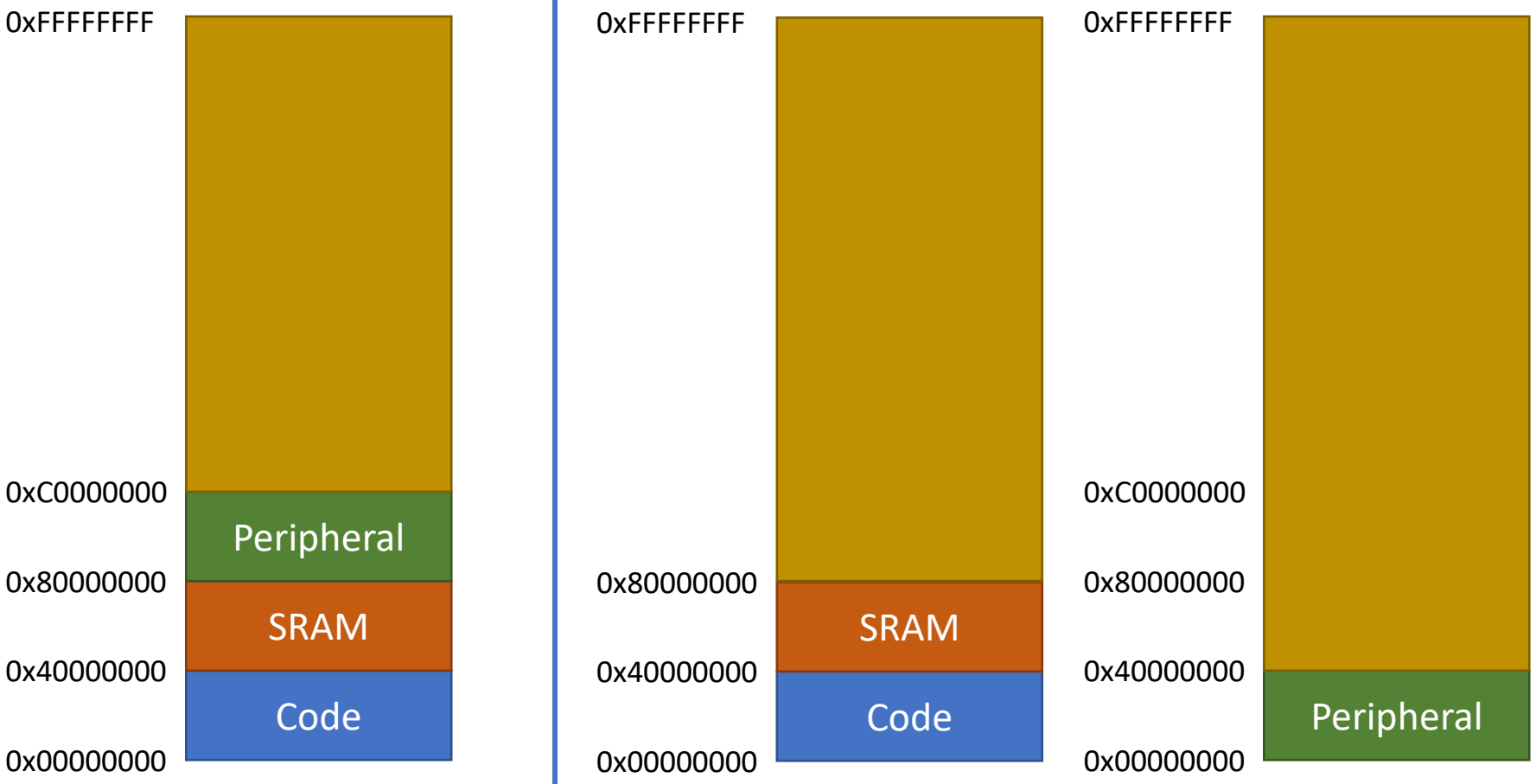

Semiconductor Memory
Nonvolatile Memory
Volatile Memory
Vendor-Programmed
User-Programmable
Programmable in Programming Unit
In-System Programmable
Static Memory
Dynamic Memory
Non-Erasable
UV-Erasable
Electrically Erasable
ROM Read Only Memory
PROM Program-mable ROM
EPROM Erasable PROM
Flash Flash EPROM
EEPROM Electrical EPROM
SRAM Static RAM
DRAM Dynamic RAM

주변장치 (Peripheral)

- 데이터 전송
  - Memory-mapped IO
  - Isolated IO
- 장치 접근
  - Polling IO
  - Interrupt-driven IO
- 제어 방식
  - Program-controlled IO
  - DMA (Direct Memory Access)



Memory-mapped IO 및 Isolated IO



TC275 Core Overview

User's Manual V2.2 2014-12

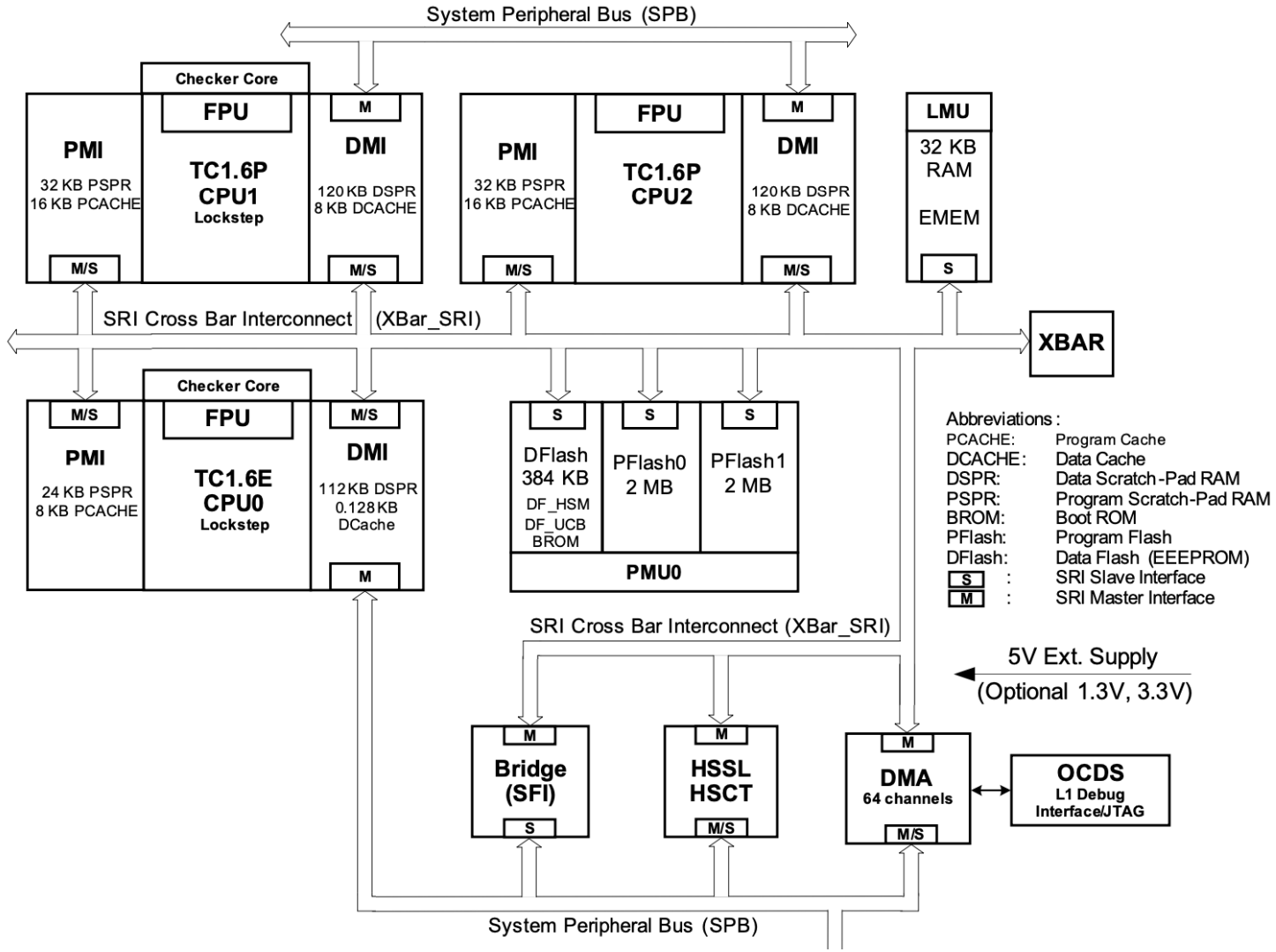
https://github.com/ace-knu/embedded

ACE Lab.

KNU

1. TC27x Introduction

SRI: Shared Resource Interconnect
SFI: SRI, FPI(Flexible Peripheral Interconnect)



Next Page

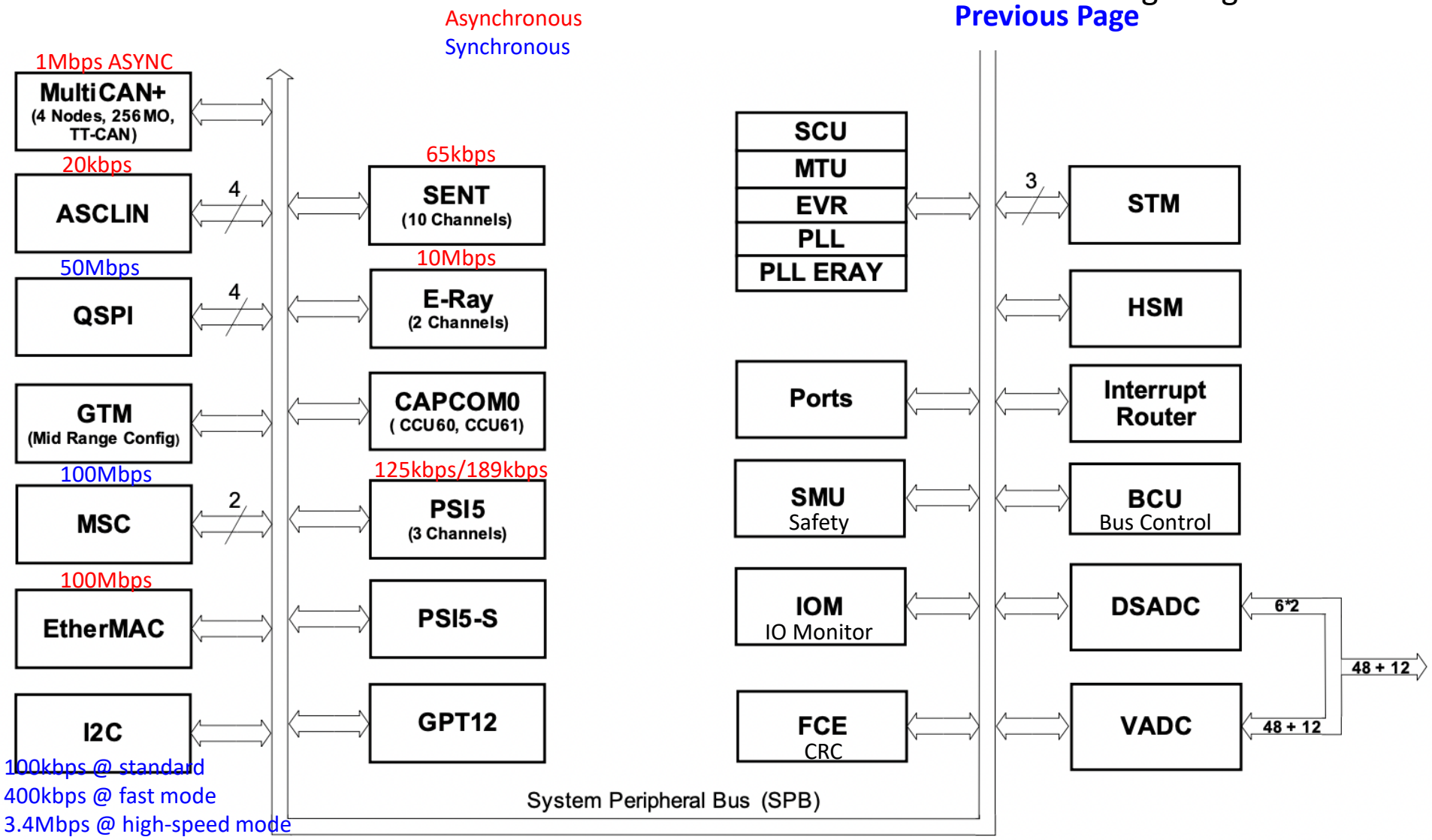
ACE Lab.

KNU

1. TC27x Introduction

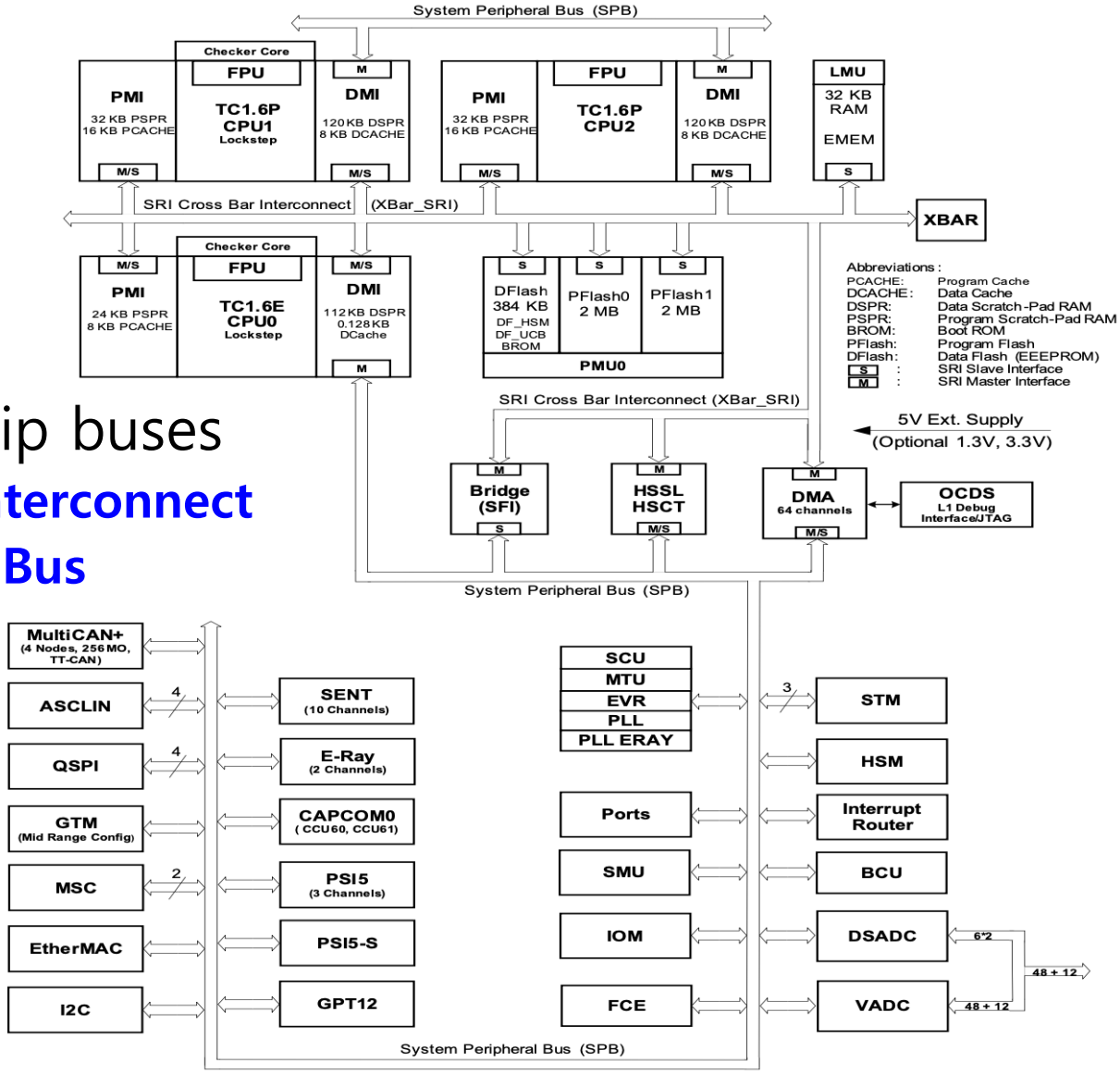
SCU: System Control Unit
MTU: Memory Test Unit
EVR: Embedded Voltage Regulator

Previous Page



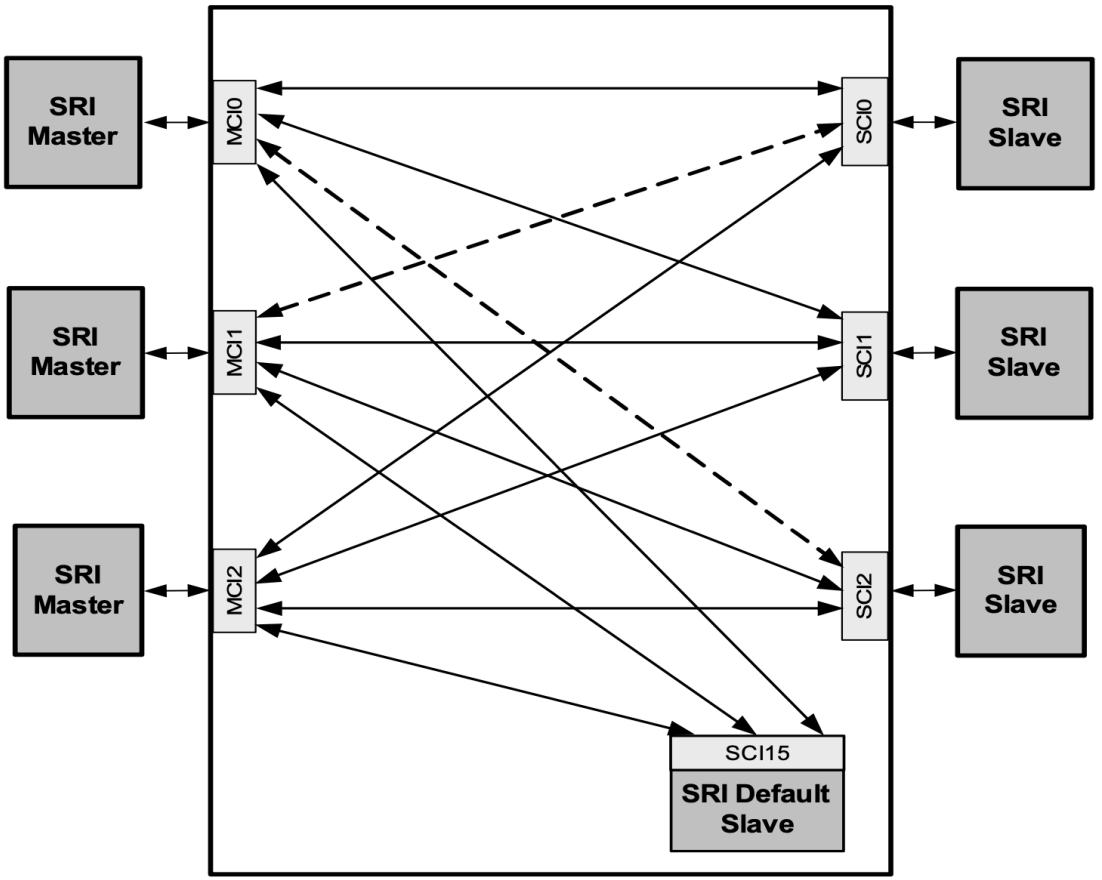
2. On-Chip System Buses and Bus Bridges

- Two independent on-chip buses
  - **SRI: Shared Resource Interconnect**
  - **SPB: System Peripheral Bus**



2. On-Chip System Buses and Bus Bridges

- SRI Crossbar (XBar_SRI)



XBar_SRI point to point connection scheme

3. Memory Maps

- Program Memory Unit (PMU0)
  - 4 MB of Program Flash Memory (PFLASH)
  - Data Flash Memory (DF_EEPROM)
  - User Configuration Blocks (DF_UCB)
  - 32 KB of Boot ROM (BROM)
- CPU0
  - 24 KB of Program Scratch-Pad SRAM (PSPR)
  - 112 KB of Data Scratch-Pad SRAM (DSPR)
  - 8 KB of Program Cache (PCache)
- CPU1 & CPU2
  - 32 KB of Program Scratch-Pad SRAM (PSPR)
  - 120 KB of Data Scratch-Pad SRAM (DSPR)
  - 16 KB of Program Cache (PCache)
  - 8 KB of Data Cache (DCache)
- LMU: 32 KB SRAM (LMURAM)

Address Map of the On Chip Bus System

Segment	Address Ragne	Size	Description
0-4	0000 0000h - 4FFF FFFFh	-	Reserved
5	5000 0000h - 5FFF FFFFh	-	CPU2 Area
6	6000 0000h - 6FFF FFFFh	-	CPU1 Area
7	**7000 0000h - 7001 BFFFh**	**112 KB**	**CPU0 Data Scratch-Pad SRAM (CPU0.DSPR)**
	7010 0000h - 7010 5FFFh	24 KB	CPU0 Program Scratch-Pad SRAM (CPU0.PSPR)
	7010 6000h - 7010 7FFFh	8 KB	CPU0 Program Cache SRAM (CPU0.PCache)
	701C 0000h - 701C 0BFFh		CPU0 Program Cache TAG SRAM (CPU0.PTAG)
8	**8000 0000h - 801F FFFFh**	**2 MB**	**Program Flash 0 (PF0)**
	8020 0000h - 803F FFFFh	2 MB	Program Flash 1 (PF1)
	8FFF 80000h - 8FFF FFFFh	32 KB	Boot ROM (BROM)
9	9000 0000h - 9000 7FFFh	32 KB	LMU SRAM (LMUSRAM)

Reserved 제외, User's Manual Table 3-2 참조

4. TC27x BootROM Content

- BOOT_TC27X
  - Startup software (SSW)
  - Software modules implementing additional functions (Bootstrap Loaders)
  - Test Firmware

Startup Software (SSW)

- SSW는 칩이 리셋 된 후 실행되는 첫 번째 소프트웨어임
- SSW는 CPU0에서 실행
  - 다른 CPU는 부팅 동안 Halt-state 유지하다 사용자 SW에 의해 시작됨
  - BootROM의 SSW 시작 주소는 CPU0의 PC 레지스터의 리셋 값임. 이 위치에서 명령어를 가져오며 장치가 시작된 후 실행되는 첫 번째 명령어임
  - 진입점 직후 펌웨어는 테스트 모드를 체크하고, 만약 선택되어 있다면 테스트 펌웨어로 점프가 실행됨
  - 마지막 SSW 명령어는 첫 번째 사용자 코드 명령어로 점프를 수행함. 첫 번째 사용자 명령어는 사용자가 선택한 스타트업 설정에 따라 다른 위치에서 가져올 수 있음
- SSW는 다음 중 하나 이상에 따라 장치를 초기화하는 절차를 포함함
  - 전용 플래시 위치에 저장된 이전 정보
  - 전용 레지스터/메모리 위치에 특수 비트/필드의 현재 상태
  - SSW 실행을 트리거한 이벤트 유형 (마지막 리셋 이벤트)
  - 외부(구성)핀에 적용된 값 (옵션)

5. CPU Subsystem

- Key CPU Features
  - 32-bit load store architecture
  - 4 GB address range
  - 16-bit & 32-bit instructions for reduced code size
  - Data types
    - Boolean, integer with saturation, bit array, signed fraction, character, double-word integers, signed integer, unsigned integer, IEEE-754 single-precision floating point
  - Data formats
    - Bit, byte (8-bit), half-word (16-bit), word (32-bit), double-word (64-bit)
  - Byte and bit addressing
  - Little-endian byte ordering for data, memory and CPU registers
  - Multiply and Accumulate (MAC) instructions: Dual 16x16, 16x32, 32x32
  - Saturation integer arithmetic
  - Packed data

5. CPU Subsystem

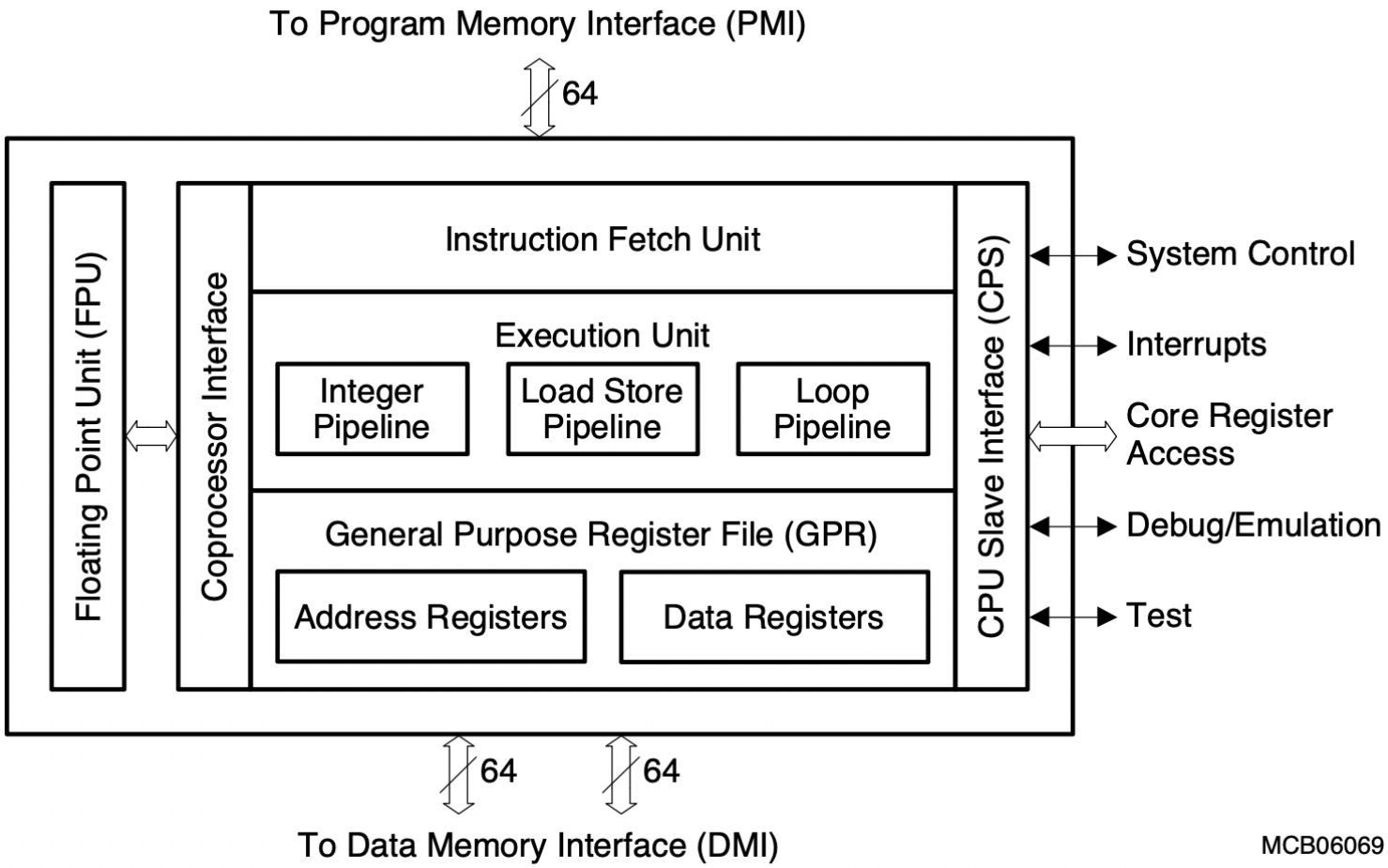
- Key CPU Features
  - Addressing modes
    - Absolute, circular, bit reverse, long + short, base + offset with pre- and post-update
  - Instruction types
    - Arithmetic, address arithmetic, comparison, address comparison, logical, MAC, shift, coprocessor, bit logical, branch, bit field, load/store, packed data
  - General Purpose Register Set (GPRS)
    - Sixteen 32-bit data registers (D0 - D15)
    - Sixteen 32-bit address registers (A0 - A15)
    - Three 32-bit status and program counter registers (PSW, PC, PCXI)
  - Flexible memory protection system providing multiple protection sets with multiple protection ranges per set
  - Temporal protection system allowing time bounded real time operation

5. CPU Subsystem

- Key CPU Features
  - Most instructions executed in 1 cycle
  - Branch instructions in 1, 2, or 3 cycles (using dynamic branch prediction)
  - Wide memory interface for fast context switch
  - Automatic context save-on-entry and restore-on-exit for: subroutine, interrupt, trap
  - Four memory protection register sets
  - Dual instruction issuing (in parallel into Integer Pipeline and Load/Store Pipeline)
  - Third pipeline for loop instruction only (zero overhead loop)
  - Single precision Floating Point Unit (IEEE-754 Compatible)
  - Dedicated integer divide unit
  - Implementation optimized for performance
  - 16 data protection ranges, 8 code protection ranges

TC1.6P Implementation Overview

- CPU block diagram



Registers

- Program Status Word Register

PSW

Program Status Word Register (CSFR_Base + $FE04_H$) **Reset Value: 0000 $0B80_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C or FS	V or FI	SV or FV	AV or FZ	SAV or FU	FX	RM		0							
rwh	rwh	rwh	rwh	rwh	rwh	rw		r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	S	PRS		IO		IS	GW	CDE	CDC						
r	rwh	rwh		rwh		rwh	rwh	rwh	rwh						

Tri Core Instruction Set Architecture

1.2.1 16-bit Opcode Formats

Note: Bit[0] of the op1 field is always 0 for 16-bit instructions.

Table 1-5 16-bit Opcode Formats

	15-14	13-12	11-10	09-08	07-06	05-04	03-02	01-00
SB	disp8				op1			
SBC	const4		disp4		op1			
SBR	s2		disp4		op1			
SBRN	n		disp4		op1			
SC	const8				op1			
SLR	s2		d		op1			
SLRO	off4		d		op1			
SR	op2		s1/d		op1			
SRC	const4		s1/d		op1			
SRO	s2		off4		op1			
SRR	s2		s1/d		op1			
SRRS	s2		s1/d		n	op1		
SSR	s2		s1		op1			
SSRO	off4		s1		op1			

1.2.2 32-bit Opcode Formats

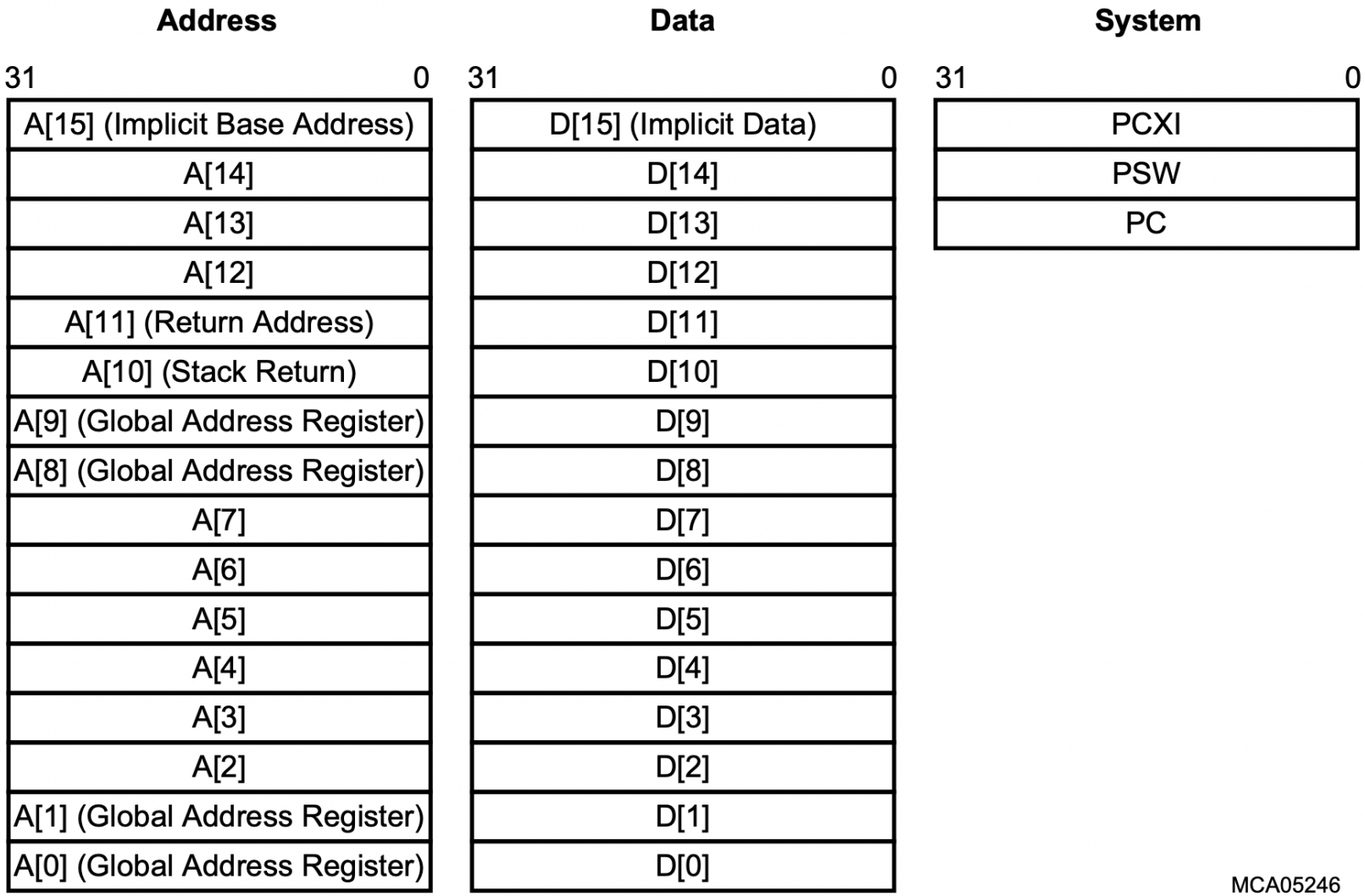
Note: Bit[0] of the op1 field is always 1 for 32-bit instructions.

Table 1-6 32-bit Opcode Formats

	31:30	29:28	27:26	25:24	23:22	21:20	21:20 19:18	17:16	15:14	13:12	11:10	9-8	7-6	5-0
ABS	off18 [9:6]		op2	off18 [13:10]		off18[5:0]			off18 [17:14]		s1/d		op1	
ABSB	off18 [9:6]		op2	off18 [13:10]		off18[5:0]			off18 [17:14]		b	bpos3	op1	
B	disp24[15:0]								disp24[23:16]				op1	
BIT	d		pos2		op2	pos1			s2		s1		op1	
BO	off10[9:6]		op2			off10[5:0]			s2		s1/d		op1	
BOL	off16[9:6]		off16[15:10]			off16[5:0]			s2		s1/d		op1	
BRC	op2	disp15							const4		s1		op1	
BRN	op2	disp15							n[3:0]		s1		n[4]	op1
BRR	op2	disp15							s2		s1		op1	
RC	d		op2			const9					s1		op1	
RCPW	d		pos		op2	width			const4		s1		op1	
RCR	d		s3		op2	const9					s1		op1	
RCRR	d		s3		op2	-			const4		s1		op1	
RCRW	d		s3		op2	width			const4		s1		op1	
RLC	d		const16								s1		op1	
RR	d		op2				-	n	s2		s1		op1	
RR1	d		op2					n	s2		s1		op1	
RR2	d		op2						s2		s1		op1	
RRPW	d		pos		op2	width			s2		s1		op1	
RRR	d		s3		op2		-	n	s2		s1		op1	
RRR1	d		s3		op2			n	s2		s1		op1	
RRR2	d		s3		op2				s2		s1		op1	
RRRR	d		s3		op2	-			s2		s1		op1	
RRRW	d		s3		op2	width			s2		s1		op1	
SYS	-		op2			-					s1/d		op1	

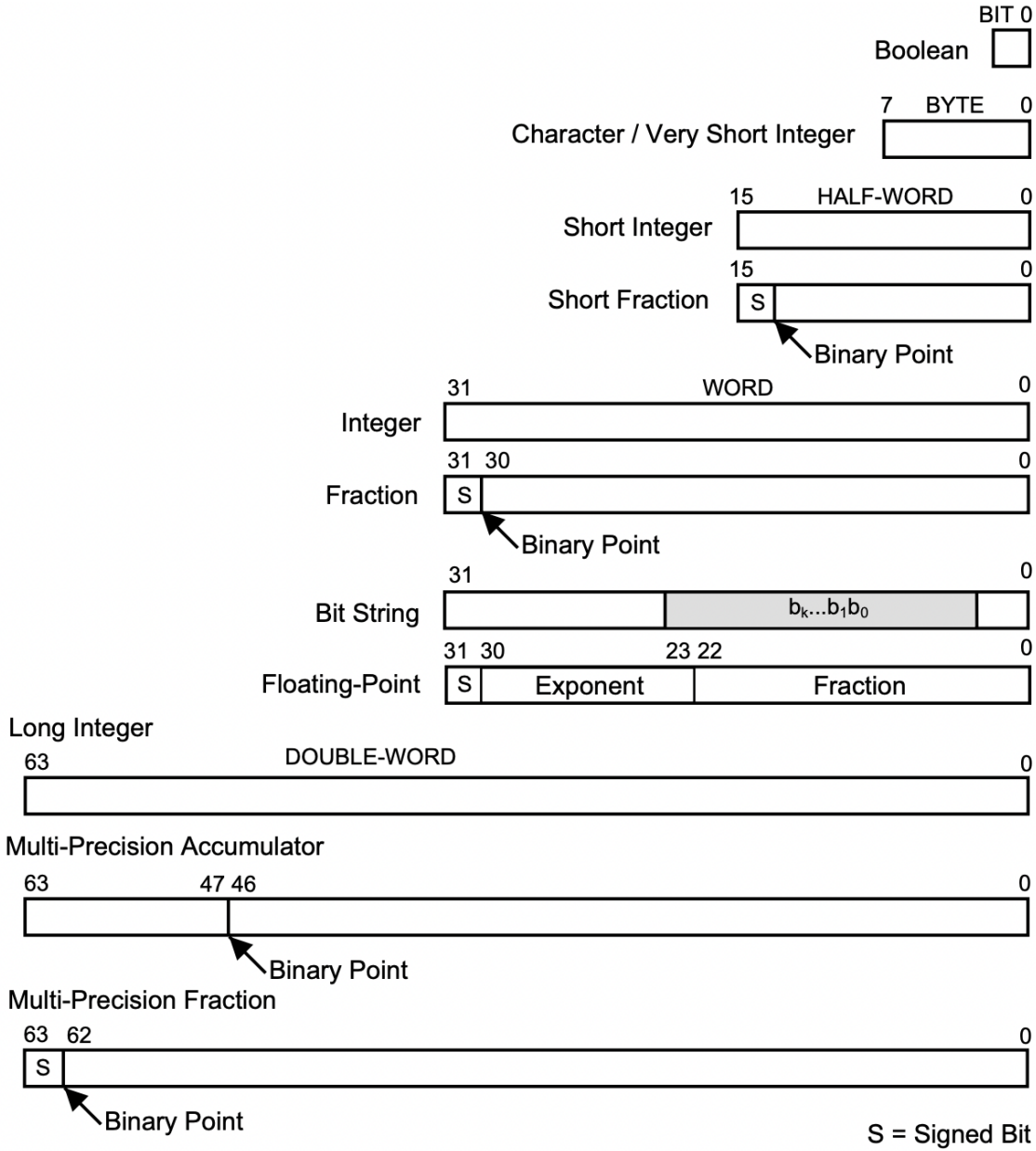
ACE Lab.

Registers



MCA05246

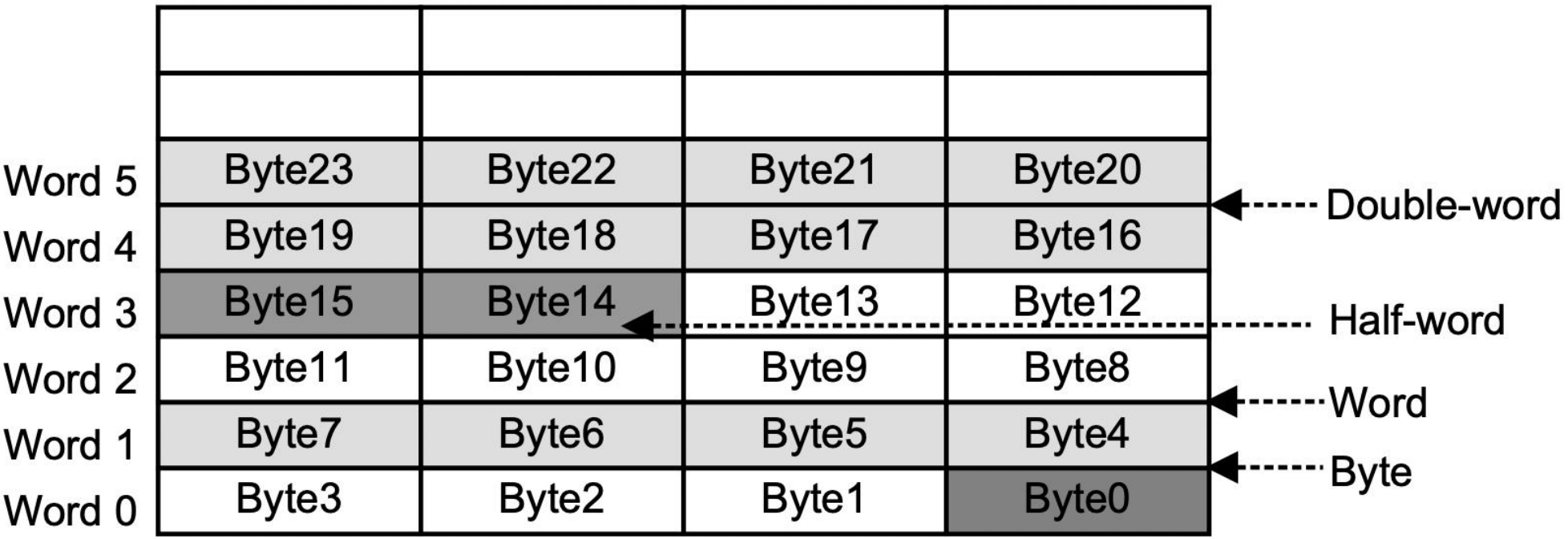
Supported Data Formats



Alignment Rules

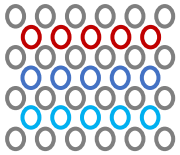
Access type	Access size	Alignment of address in memory
Load, Store Data Register	Byte	Byte (1_H)
	Half-Word	2 bytes (2_H)
	Word	2 bytes (2_H)
	Double-Word	2 bytes (2_H)
Load, Store Address Register	Word	4 bytes (4_H)
	Double-Word	4 bytes (4_H)
SWAP.W, LDMST	Word	4 bytes (4_H)
ST.T	Byte	Byte (1_H)
Context Load / Store / Restore / Save	16 x 32-bit registers	64 bytes (40_H)

Byte Ordering



Q & A

Thank you for your attention



Architecture and Compiler
for Embedded Systems Lab.

School of Electronics Engineering, KNU

ACE Lab. (jcho@knu.ac.kr)

ACE Lab.

KNU