Mittuniversitetet
MID SWEDEN UNIVERSITY

Försättsblad tentamen / Examination cover

Anonymitetskod / Anonymous code		
0 1 2 - 0 6		
Kurskod / Course code E T 0 1 4 G	Provkod/ Test code T 1 0 4	Tentamensdatum / Examination date 2 0 1 7 - 0 3 - 1 7
Kursnamn / Course name Elektroteknik GR (B), Programmering av inbyggda system		
Provnamn / Test name Skriftlig tentamen, teori		

Skriv din anonymitetskod på varje inlämnat papper
Write your anonymous code on each sheet submitted

Sätt ett kryss (x) för varje inlämnad uppgift
Use an x to indicate which questions has been submitted

Markera nedan med X / Mark below with an X		Poäng / Credit	Lärarens anteckningar / Teacher's notes	Markera nedan med X / Mark below with an X		Poäng / Credit	Lärarens anteckningar / Teacher's notes
1	X	1,5		16			
2	X	5		17			
3	X	4		18			
4	X	2		19			
5	X	2		20			
6	X	2		21			
7	X	3.5		22			
8	X	2		23			
9	X	2		24			
10	X	5		25			
11	X	1.5		26			
12	X	2		27			
13	X	2		28			
14	X	2		29			
15	X	2		30			
Poängsumma / Points		38.5	Betyg / Grade: A	Lärarsign./ Teachers sign			Peng Cheng

Fylls i av tentamensvakt / To be filled in by the invigilator

Antal lösa blad/ No. of sheets submitted	Inlämnad tentamen / Submitted exam	Leg kontroll / Control identification	Sign. tentamensvakt / Sign. invigilator
15	1	ok	GD

Försättsbladet skall alltid lämnas in även om ingen uppgift behandlats
Examination cover should always be submitted even if no questions are answered

7313318589

Mittuniversitetet
MID SWEDEN UNIVERSITY

Anonymitetskod/ Anonymus code: 0 1 2 - 0 6

Sidnr / Page no: 1

1) 32 bit ~~fixed~~ ~~floating~~-point multiplication is faster because it only uses integer operation hardware, which is faster and ~~as~~simpler. However, floating-point multiplication needs more complex operations in hardware.

Lärarens anteckning / Teachers note:

①

② Instruction 1:
assembly syntax: mov R4, (1101 1100 1011 1010)$_2$ or mov R4, DCBA
R4 is modified, and its new value is ~~0x0000~~0xDCBA.
No status flags have been modified.

③ Instruction 2:
assembly syntax: ~~mov R4~~ orh R4, (1010 1011 1100 1101)$_2$ or orh R4, ABCD
R4 is modified, and its new value is 0xABCDDCBA.
~~Status flags N and Z are~~ Status flag N is modified. The new value of N is 1.
~~The new value of Z is 0. The value of Z is 0.~~ The value of Z is 0.

④ Instruction 3:
assembly syntax: mov R3, (1010 0001 0011 1010)$_2$ or mov R3, A11A.
R3 is modified, and its new value is 0x0000A11A.
No status flags have been modified.

⑤ Instruction 4:
assembly syntax: orh R3, (0000 1011 1011 0000)$_2$ or orh R3, 0BB0.
R3 is modified, and its new value is 0x0BB0A11A.
Status flags N is modified. The new value of N is 0.
The value of Z is 0.

⑥ Instruction 5:
assembly syntax: add R5, R3, R4<<2
R5 is modified, and its new value is 0x~~BAE81402~~ BAE81402.
Status flag N is modified. The new value of N is 1.

⑦ Instruction 6:
assembly syntax: ~~①~~ eor R3, R4, R5>>2.
R3 is ~~modified~~ modified, and its new value is 0x85770BA
No status flag has been modified. The value of N is still 1.

Lärarens anteckning / Teachers note:

3) "Volatile Memory" means the memory loses data after the power is switched off.

"Non Volatile Memory" means the memory can retain data after the power is switched off.

Volatile memory: SRAM. SDRAM.

"Non-volatile memory" is used to store program; thanks ~~for the~~ to two main properties:

1) It can still retain program after the power is switched off.

2) The speed of reading non-volatile memory is faster than writing it.

Lärarens anteckning / Teachers note:

Lärarens anteckning / Teachers note:

4) 32kHz oscillator is slower.

RC oscillator is completely internal in the microprocessor.

PLL module is needed to be enabled because it can amplify the freqeuncy of crystal oscillator with the digital circuit inside.

5). ① Round-Robin Arbitration

② Fixed Priority Arbitration

"Fixed Default Master" is best to use to minimize the access latency.

Lärarens anteckning / Teachers note:

Lärarens anteckning / Teachers note:

b). Local Bus Interface

It is faster because it connects the CPU and GPIO pins directly and serves as a "shortcut" for their communication. There is no PBA involved during the process.

Lärarens anteckning / Teachers note:

7) ① USART can communicate with external devices without a clock pin connection.

② TWI has the slowest maximum communication data rate with external devices

③ EIC and PM are needed to wake up the microprocessor from the deepest sleep mode.

④ TC can measure the duty cycle of a external waveform.

⑤ RTC can keep tracking of the longest time interval.

⑥ PWM can generate a digital waveform with configurable duty cycle.

⑦ PM can set the operating frequency of cpu.

⑧ PDCA and SPI can communicate with a SD card.

Lärarens anteckning / Teachers note:

8) PDCA is ~~used for~~ "DMA controller", and it is used for transmitting data between main memory and external devices(Disk, SD card and so on)

It is recommended because it can decreases the times of I/O ~~interrupt~~ interruption. Thereby, it frees CPU to ~~fini~~ deal with other tasks to CPU such as calculation. The efficiency of CPU can improve a lot.

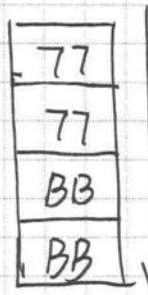
9) A pointer is an object referring to another variable stored elsewhere with its address.

4 bytes are needed in memory for a point definition.

Lärarens anteckning / Teachers note:

Lärarens anteckning / Teachers note:

10) In AT32UC3A, the data are stored in "Big-endian" way.



X = 0x7777BBBB

y[0] = 0x7777

y[1] = 0xBBBB

① "z.x|=mask_1" means "z.x=z.x|mask_1". The bitwise OR is operated between z.x and mask_1, and the result is stored back in z.x.

Result: X = 0x7777BFFB ①. y[0]=0x7777, y[1]= 0xBFFB.

② "z.y[1]^=MASK_2" means "z.y[1]=zy[1]^MASK_2". The bitwise Exclusive-OR is operated between z.y[1] and MASK_2, and the result is stored back in z.y[1].

Result: X= 0x7777B99B, y[0]=0x7777, y[1]=0xB99B

③ "z.y[0]<<=3" means "z.y[0]=z.y[0]<<3", The binary bits of z.y[0] logically left-shift; and the result is stored back in y[0].

Result: X=0xBBB8B99B, y[0]=0xBBB8, y[1]=0xB99B.

④ if(z.x&(1UL<<30)) judges whether the 30th bit of z.x is 1. If the 30th bit of z.x is 1, execute the following expression.

z.x = 0xBBB8B99B, so the 30th bit is 0. The condition is not satisfied.

"z.x=~z.x" means opposite every bit of z.x. However, "z.x=~z.x" won't be executed.

Result: X=0xBBB8B99B, y[0]=0xBBB8, y[1]=0xB99B.

Lärarens anteckning / Teachers note:

11) unsigned char: 0 ~ ~~127~~255
(0 = minimum, 255 = maximum)

signed char: ~~-256 ~ 255~~
-128 ~ 127
(-128 = minimum, 127 = maximum)

unsigned short int: 0 ~ 65536
(0 = minimum, 65536 = maximum)

signed short int: -32768 ~ 32767
(-32768 = minimum, 32767 = maximum)

The maximum of unsigned char +1 is 0.
The maximum of signed char +1 is -128.
The maximum of unsigned short int +1 is 0
The maximum of signed short int +1 is -32768

The minimum of unsigned char -1 is 255
The minimum of signed char -1 is 127.
The minimum of unsigned short ~~char~~ int ~~is~~ -1 is 65536.
The minimum of ~~u~~signed short int -1 is 3267.

Lärarens anteckning / Teachers note:

12) "Big-Endian" means "The most significant byte data are stored in the locations whose addresses are lowest.

"Little-Endian" means that the least significant byte data are stored in the locations whose addresses are lowest.

AT32UC3A uses "Big-endian".
Intel X86 processor uses "Little-endian".

If these two processors communicate, the disorder of bytes might ~~appe~~happen (if the size of data is ~~long~~ bigger than 1 byte)

For example,
"0x12345678" in "Little-endian" machine may be interrepted as "0x78563412", which could give rise to some problems.

Lärarens anteckning / Teachers note:

B "Local variable" in C could have its memory location shared for other use.

"Global variable" will be automatically initialized. to 0 by the compiler before its use.

Lärarens anteckning / Teachers note:

14) "0" is logical FALSE.

Numbers other than "0" are logical TRUE.

"while(!(-1));" can be regarded as "while(FALSE);", so it actually means "nothing to do" in C.

Lärarens anteckning / Teachers note:

15). Only one.

Becuse in ATmega32A microprocesser, there are ~~stages~~ 3 stages of pipelines. They are "Fetch", "Instruction Decode" and "Instruction Execution". In this way, it can only execute one instruction at ~~the~~ the stage of "execution" at any given time.

012-06

Exam in "Programming of Embedded Systems"

Course: "Programming of Embedded Systems": ET014G
Date: 17th of March, 2017
Examiner: Peng Cheng Office Tel: 010-1428495
Requirement: write **very clear letters** on paper, write **only one question** per page!!!
Allowed to use: Pen (**not pencil!**), eraser, ruler. **Pocket calculator is not allowed**!
Marks: Each question has a different mark.

Number of marks (p)	Grade
>=36	A
>=32	B
>=28	C
>=24	D
>=20	E (Approved) ☺
<20	F

~~E 10~~

1) In AT32UC3A microprocessor, which execution is faster, the 32bit fixed-point multiplication or the 32bit floating-point multiplication? And why? (2p)
2) After reading the reference material on the different instruction formats, write in the **assembly syntax** what the following AVR32 binary program is doing, and what **registers** and **status flags** have been modified and what are their new **values** after each instruction. (6p)

AVR32 binary program:

```
Instruction 1: 0xE064DCBA
Instruction 2: 0xEA14ABCD
Instruction 3: 0xE063A11A
Instruction 4: 0xEA130BB0
Instruction 5: 0xE6040025
Instruction 6: 0xE9E52223
```

E 1110 A 1010
F 1111 B 1011
C 1100
D 1101

3) What are the meanings of "volatile memory" and "non-volatile memory"? And for the different memories available in **AT32UC3A microprocessor and EVK1100 board**, give at least **two** different examples of "volatile memory". And which memory is typically used to store program and what are its properties that make it suitable for storing program but not for storing data? (4p)
4) In AT32UC3A microprocessor, there are RC oscillator and 32KHz oscillator, which one is slower? And which one is completely internal in the microprocessor? And in order to run the CPU at 66MHz, except for the main crystal oscillator, which module also need to be enabled? And why? (2p)
5) In HMATRIX of AT32UC3A microprocessor, what are the names of arbitration mechanisms used to solve the conflict of multiple masters trying to access single slave? And in order to minimize the access latency between the CPU instruction master and the slave module which store application program, which of the following settings is best to use? "No Default Master", "Last Accessed Default Master", "Fixed Default Master". (2p)
6) What is the fastest way (which hardware interface to use) for CPU to access the registers of the GPIO controller in the AT32UC3A microprocessor? Why is this way faster than the conventional way of accessing the registers of the GPIO controller? (2p)
7) In AT32UC3A microprocessor, there are PM, TC, USART, INTC, RTC, EIC, SPI, ADC, TWI, PDCA and PWM modules. Which module can communicate with external devices without a clock pin connection? Which module has the slowest

maximum communication data rate with external devices? Which **two** modules are needed to wake up the microprocessor from the deepest sleep mode? Which module can measure the duty cycle of an external digital waveform? Which module can keep tracking of the longest time interval? Which module can generate a digital waveform with configurable duty cycle? Which module can set the operating frequency of CPU? Which **two** modules can communicate with a SD card?(4p)

8) In AT32UC3A microprocessor, what is PDCA module used for? And why is using it recommended for large amount of I/O data? (2p)
9) What is a pointer variable in C? In AT32UC3A microprocessor, how many bytes are needed in memory for a pointer definition? (2p)
10) Describe what this C code is doing in detail **in each line of code** and calculate the **result in each line of code** for x and y: (6p)

```
extern volatile mask_1 = 0x00000CC0;
#define MASK_2    0x0660
Union
{
U32 x = 0x7777BBBB;
U16 y[2];
} z;
z.x |= mask_1;
z.y[1] ^= MASK_2;
z.y[0] <<= 3;
if (z.x & (1UL << 30))
{
      z.x = ~z.x;
}
```

11) What are the maximum and minimum numbers represented in an "unsigned char", a "signed char", an "unsigned short int" and a "signed short int"? What are the results of the maximum numbers of these four variables + 1? What are the results of the minimum numbers of these four variables - 1? (2p)
12) What do "big-endian" and "little-endian" mean? And what are the endianness used in an AT32UC3A microprocessor and an Intel X86 processor? And what problem could this bring if these two processors communicate?(2p)
13) For the "global variable" and "local variable" in C, which variable could have its memory location shared for other use? Which variable will be automatically initialized to 0 by the compiler before its use? (2p)
14) What decimal numbers are logical FALSE and TRUE represented in C programming language? What do you think "while(!(-1));" will be treated in C? (2p)
15) In the AT32UC3A microprocessor, how many instructions can it execute at any given time? And why? (2p)

The following is the reference material for question 2.

MOV – Move Data Into Register

Description

Moves a value into a register. The value may be an immediate or the contents of another register. Note that Rd may specify
PC, resulting in a jump. All flags are unchanged.

Operation:

I. Rd ← SE(imm8);
II. Rd ← SE(imm21);
III. Rd ← Rs;

Syntax:

I. mov Rd, imm
II. mov Rd, imm
III. mov Rd, Rs

Operands:

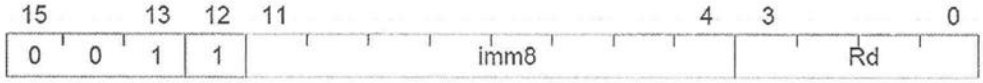
I. d ∈ {0, 1, ..., 15}
imm ∈ {-128, -127, ..., 127}
II. d ∈ {0, 1, ..., 15}
imm ∈ {-1048576, -104875, ..., 1048575}
III. d, s ∈ {0, 1, ..., 15}

Status Flags:

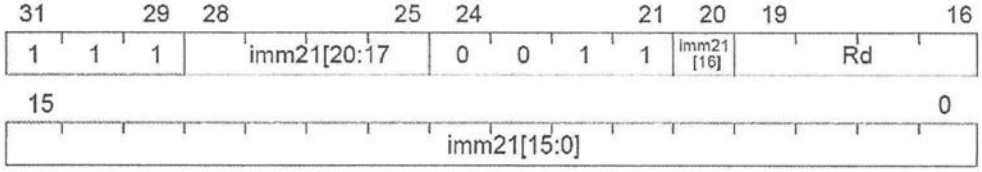
Q: Not affected.
V: Not affected.
N: Not affected.
Z: Not affected.
C: Not affected.

Opcode:

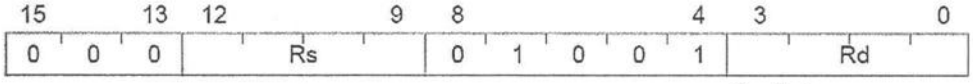
Format I:



Format II:



Format III:



1110 0000 0110 0100 1101 1100 1010 1010
R4

1110 0000 0110 0011 1010 0001 0001 1010
R3

ORH, ORL – Logical OR into high or low half of register

Description

Performs a bitwise logical OR between the high or low word in the specified register and a constant. The result is stored in
the destination register.

Operation:

I. Rd[31:16] ← Rd[31:16] ∨ imm16;

II. Rd[15:0] ← Rd[15:0] ∨ imm16;

Syntax:

I. orh Rd, imm

II. orl Rd, imm

Operands:

I, II. d ∈ {0, 1, …, 15}

imm ∈ {0, 1, ..., 65535}

Status Flags:

Q: Not affected

V: Not affected

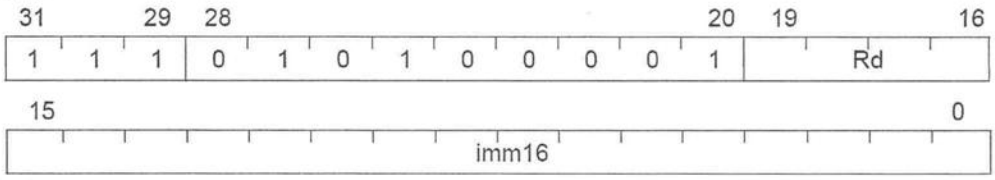
N: N ← RES[31]

Z: Z ← (RES[31:0] == 0)

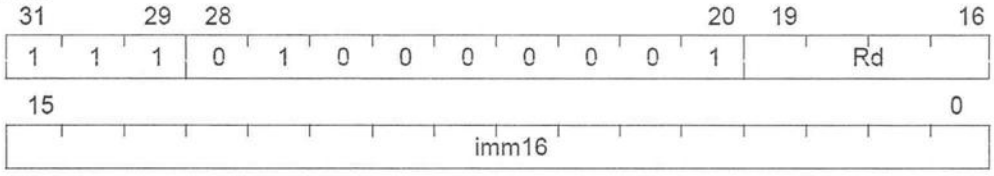
C: Not affected

Opcode:

Format I:



Format II:



1110 1010 000 0100 1010 0111 1100 1110

R4

1110 1010 000 0011 0000 1011 1011 0000

R3

R4

ADD– Add without Carry

Description

Adds the two registers specified and stores the result in destination register. Format II allows shifting of the second operand.

Operation:

I. Rd ← Rd + Rs;

II. Rd ← Rx + Ry<< sa2;

Syntax:

I. add Rd, Rs

II. add Rd, Rx, Ry << sa

Operands:

I. {d, s} ∈ {0, 1, ..., 15}

II. {d, x, y}∈ {0, 1, ..., 15}

sa ∈ {0, 1, 2, 3}

Status Flags

Format I: OP1 = Rd, OP2 = Rs

Format II:OP1 = Rx, OP2 = Ry << sa2

Q: Not affected

V: V ← (OP1[31] ∧ OP2[31] ∧ . RES[31]) ∨ (. OP1[31] ∧ . OP2[31] ∧ RES[31])

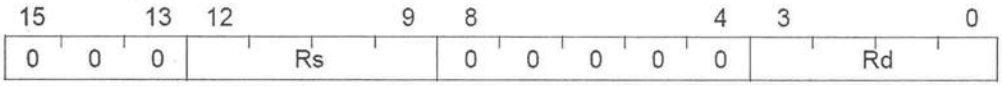
N: N ← RES[31]

Z: Z ← (RES[31:0] == 0)

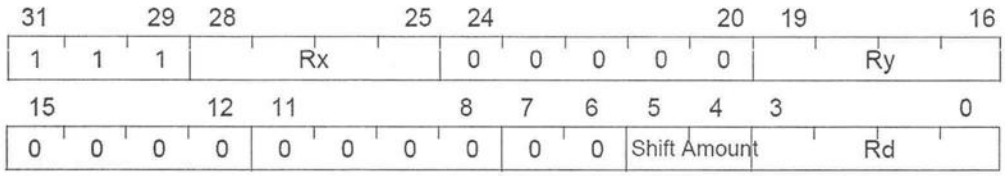
C: C ← OP1[31] ∧ OP2[31] ∨ OP1[31] ∧ . RES[31] ∨ OP2[31] ∧ . RES[31]

Opcode:

Format I:



Format II:



EOR – Logical Exclusive OR with optional logical shift

Description

Performs a bitwise logical Exclusive-OR between the specified registers and stores the result in the destination register.

Operation:

I. Rd ← Rd ⊕ Rs;

II. Rd ← Rx ⊕ Ry << sa5;

III. Rd ← Rx ⊕ Ry >> sa5;

Syntax:

I. eor Rd, Rs

II. eor Rd, Rx, Ry << sa

III. eor Rd, Rx, Ry >> sa

Operands:

I. {d, s} ∈ {0, 1, …, 15}

II, III. {d, x, y} ∈ {0, 1, …, 15}

sa ∈ {0, 1, …, 31}

Status Flags

Q: Not affected

V: Not affected

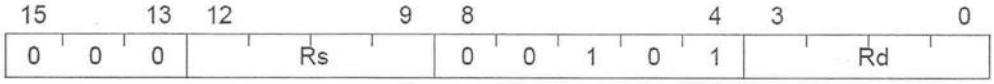
N: N ← RES[31]

Z: Z ← (RES[31:0] == 0)

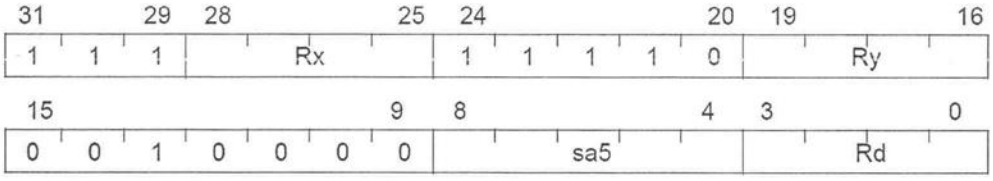
C: Not affected

Opcode:

Format I:



Format II:



Format III: