

Experiment 26

Introduction to the ARM Microprocessor

Experimental Results

Important: Marking of all coursework is anonymous. Do not include your name, student ID number, group number, email or any other personal information in your report or in the name of the file submitted via VITAL. A penalty will be applied to submissions that do not meet this requirement.

Instructions:

- Read this script carefully before attempting the experiment.
- Answer the pre-lab questions before attending the lab (worth 10%).
- Have a look at the document “Exp 26-Submission Template” available on Canvas to have an idea about the experimental results that you have to submit. Experimental results submission is worth 90% of the experiment mark.
- Keep a record of all screenshots, results, answers, comments made and graphs plotted in a logbook. When you submit your work, make sure that all the results, screenshots, etc., are clear and readable; otherwise, you’ll lose marks.
- **Important:** When taking snapshots from the screen during the simulation, for each and every snapshot, please **FIRST** use **PrintScreen** under Microsoft Windows, or **CMD+SHIFT+3** or **Screenshot** under macOS (which shows the entire desktop including date and time and helps to identify this as uniquely your work) **AND THEN ALSO** use the **Snipping tool** (Microsoft Windows) or **Screenshot** tool (macOS) to show only the relevant part of your simulation. If the entire desktop is not visible in your first screenshot, the associated ‘focussed’ screenshot will be ignored, and the corresponding section of your report will receive a mark of zero.
- Including screenshots of other people’s work is considered academic malpractice and will be penalised in accordance with the University Codes of Practice.

- All code and text should be included as text and not as screenshots.

Experimental Results and Comments (Total 90 marks):

Please provide the required screenshots, photos, code, values highlighted in the script according to the following requirements (screenshots should be clear and readable):

1. Answer to Q1 [4 marks]

Answer:

The second digit is used to identify the register used.

Explanation:

The instructions and machine codes are

220E MOVS r2, #0x0E

2325 MOVS r3, #0x25

According to the underlined numbers, it is clear that the second digit of the machine code identifies the register used.

2. Answer to Q2 [4 marks]

Answer:

The third and fourth digits give the number to be used.

Explanation:

The instructions and machine codes are

220E MOVS r2, #0x0E

2325 MOVS r3, #0x25

From the underlined numbers, it is can be seen that the last two digits identify the value.

3. Answer to Q3 [4 marks]

Answer:

The value in R4 is 0x00000015

Explanation:

The instruction operates $r4 = r2 + 7_{(10)}$. The value hold by R2 before the execution is 0x00000000E. Therefore, the value hold by r4 is 0x00000015.

4. Answer to Q4 [4 marks]**Answer:**

The order of the instruction is important.

Explanation:

For examples, by executing the instructions after the addresses 0x00000266 and 0x00000268, the first instruction operates $r1 = r2 - r3$, while the other instruction operates $r2 = r3 - r2$. If change the order, r1 is equal to $r3 - r2$ and r2 is equal to $r2 - r3$. Those different instructions can execute different results. Therefore, it is important to check the order of the instructions.

5. Answer to Q5 [4 marks]**Answer:**

These values are expected.

Explanation:

The initial value in R2 is 0x00000064, while for R3 is 0x0000000A. The following shows the instructions executed form 0x0000025E to 0x0000026A.

$$r4 = r2 + 7_{(10)} = 0x0000006B$$

$$r5 = r2 + r3 = 0x0000006E$$

$$r0 = r2 - 1_{(10)} = 0x00000063$$

$$r1 = r2 - r3 = 0x0000005A$$

$$r2 = r3 - r2 = 0XFFFFFFA6$$

It can be seen that R3 does not change. R2 changed to 0XFFFFFFA6.

6. Answer to Q6 [4 marks]**Answer:**

These values are expected.

Explanation:

The initial value in r2 is 0x000000CD, while for r3 is 0x000000AB. The following shows the instructions executed from 0x0000025E to 0x0000026A.

$$r4 = r2 + 7_{(10)} = 0x000000D4$$

$$r5 = r2 + r3 = 0x00000178$$

$$r0 = r2 - 1_{(10)} = 0x000000CC$$

$$r1 = r2 - r3 = 0x00000022$$

$$r2 = r3 - r2 = 0xFFFFFFFDE.$$

It is similar to Q5, the value held by R3 does not change. The value held by R2 changed.

7. Answer to Q7 [4 marks]

Answer:

AND logic function

Explanation:

The value in r2 is 0x00000011, while the value in r3 is 0x00000101. The result in r4 is 0x00000001.

Consider the last four digits,

Register	4 th digit	3 rd digit	2 nd digit	1 st digit
R2	0	0	1	1
R3	0	1	0	1
R4	0	0	0	1

Therefore, it is an AND logic.

8. Answer to Q8 [4 marks]

Answer:

The order is not important for other functions.

Explanation:

For AND, ORR and EOR, they consider the combination of each digit for the two values rather than the order.

For example,

ORRS r5, r5, r3 executes $r5 = r5 \text{ or } r3$.

ORRS r5, r3, r5 executes $r5 = r5 \text{ or } r3$.

These two instructions obtained the same result.

9. Answer to Q9 [4 marks]

Answer:

The program counter moves to the address at 0x00000288. Then, jump back to 0x000000286.

Explanation:

The instruction and address are

0x0000286 ADDS r2, r2, #1;

0x0000288 B 0x00000286;

For the branch instruction, it means go back to the address 0x00000286. Therefore, we can see the program complied in this loop.

10. Answer to Q10 [5 marks]

Answer:

The value is R2 changes from 0x00000000 to 0x00000005 after five cycles.

Explanation:

Register R2 works like a counter. After each time the instruction branch instruction is executed, the value in R2 will plus 1. Thus, after 5 cycles, the value is 0x00000005.

11. Answer to Q11 [4 marks]

Answer:

That is because the program complies at a certain speed. Therefore, the value hold in R2 after 40s shall be approximately twice the value in R2 after 20s.

Explanation:

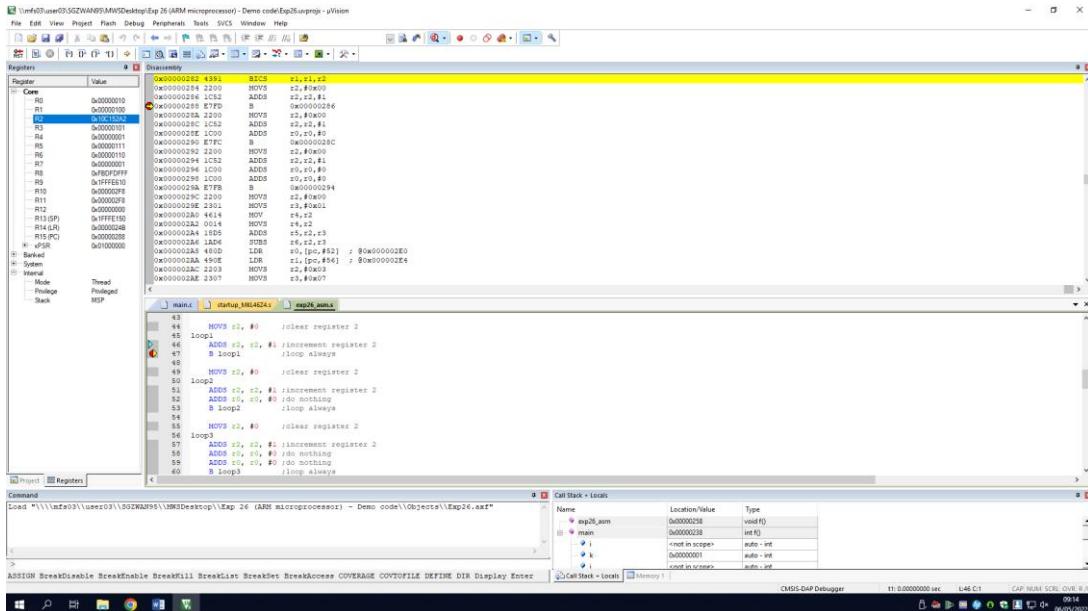


Figure 1: R₂ value

As Figure 1 shows, the value in R₂ is 0x10C152A2 after 20.15s, which is 281105058₍₁₀₎.

When the time increases to 40.35s. The value hold in R₂ is 0x21C6D323, which is 566678307₍₁₀₎. Figure 2 shows the result.

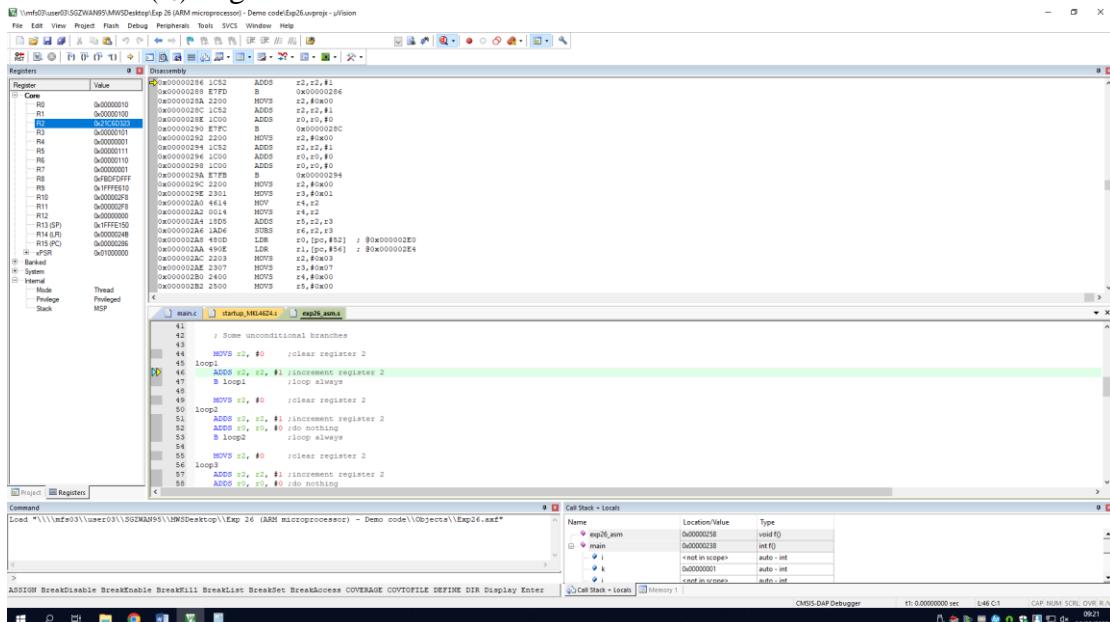


Figure 2: R₂ value

Thus, it can be concluded that the program complies at a certain speed and the value should be twice.

12. The graph of Section 9 (using MS Excel or MATLAB) [6 marks]

Screenshot (PrintScreen or Screenshot):

When counter is set to 0x00000028A

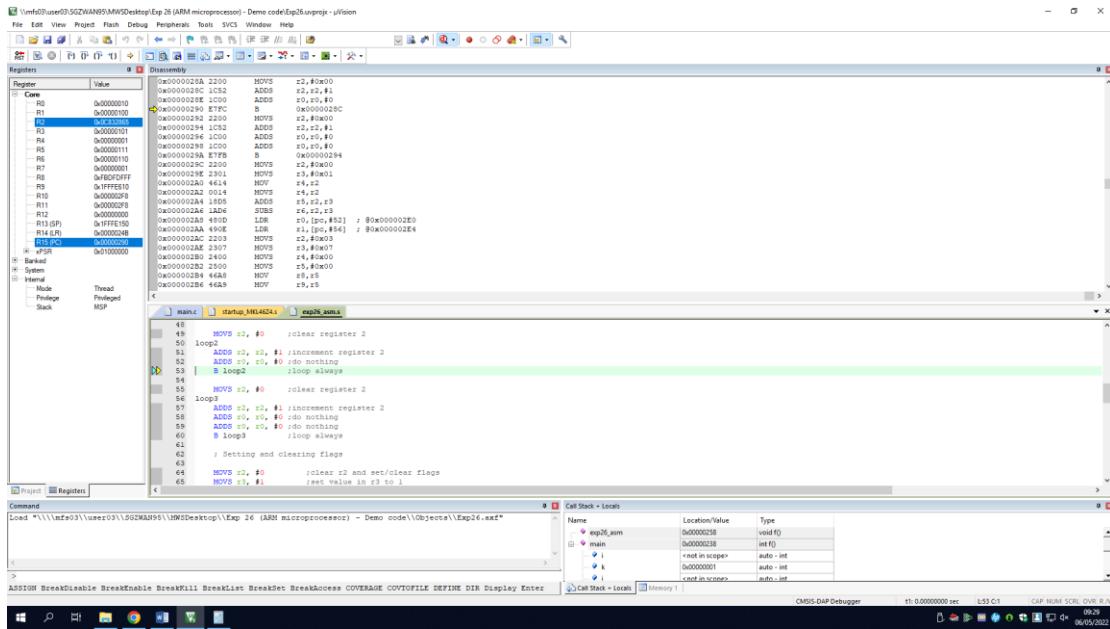


Figure 3: 0x00000028A result (20.10s)

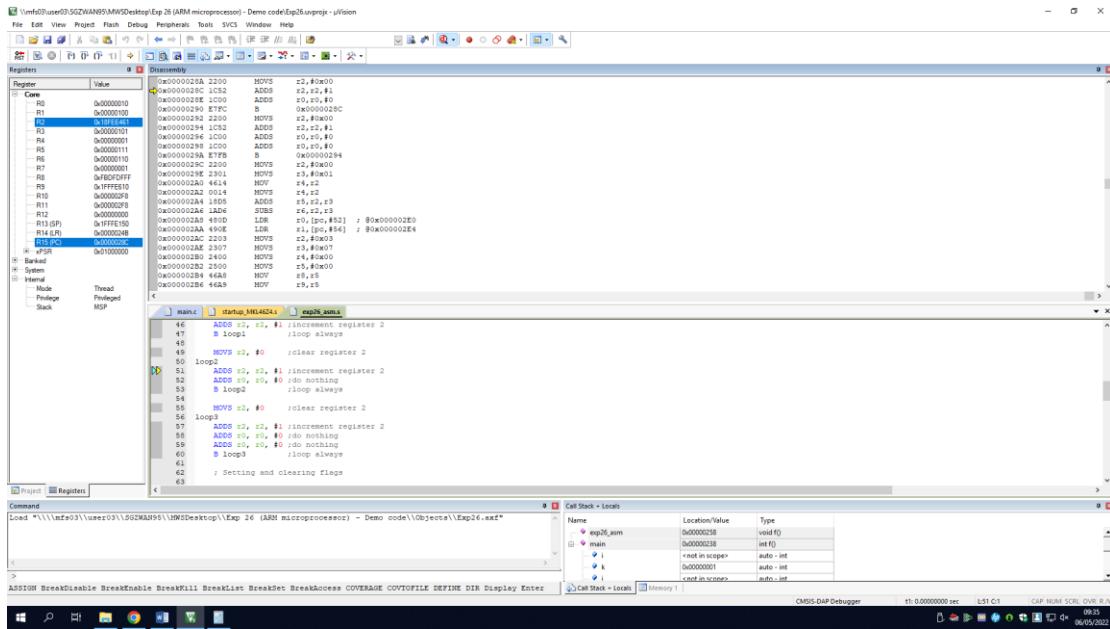


Figure 4: 0x00000028A result (40.09s)

Set counter to 0x000000292

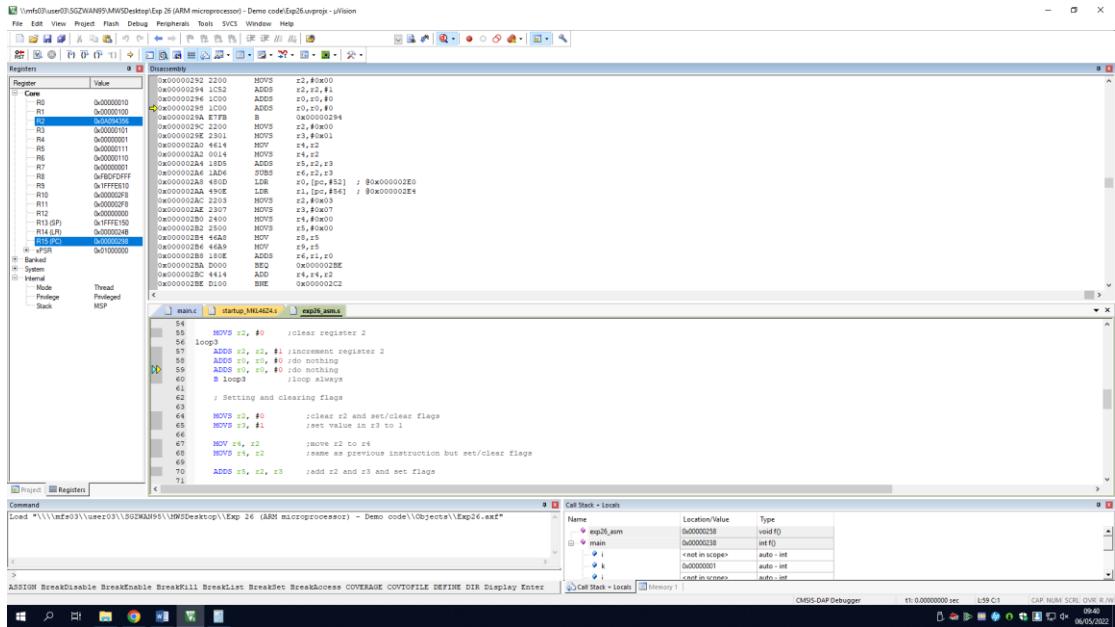


Figure 5: 0x000000292 result (20.00s)

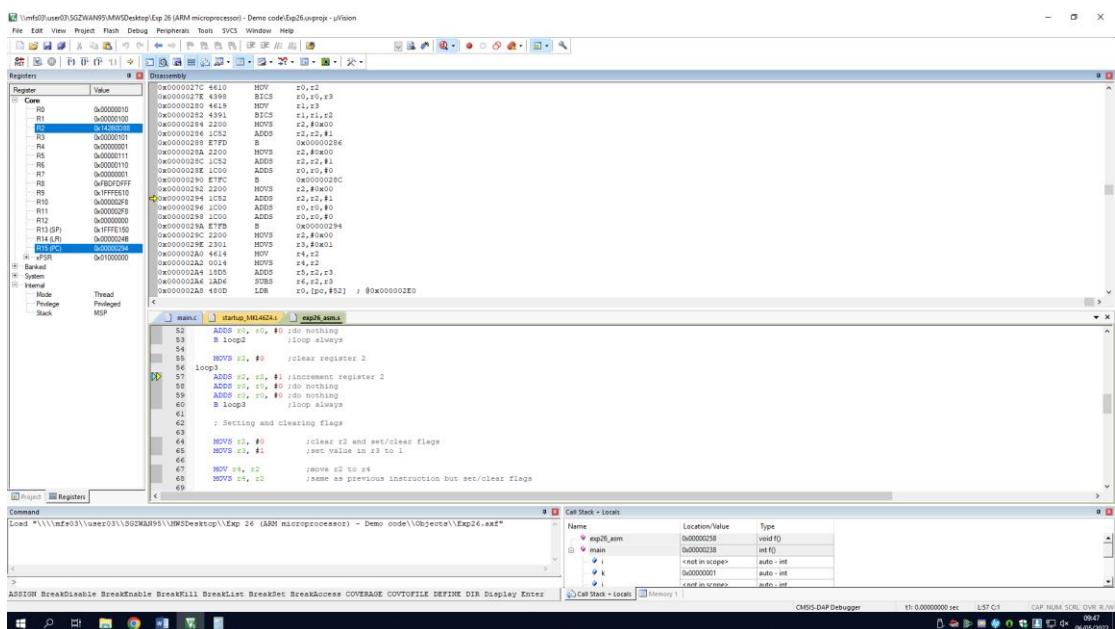


Figure 6: 0x000000292 result (40.15s)



Figure 7: plot

Screenshot (Snipping tool, Preview or equivalent):

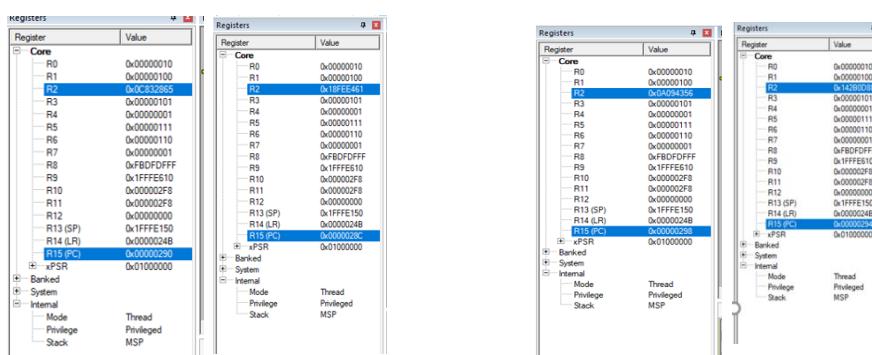


Figure 7: Registers

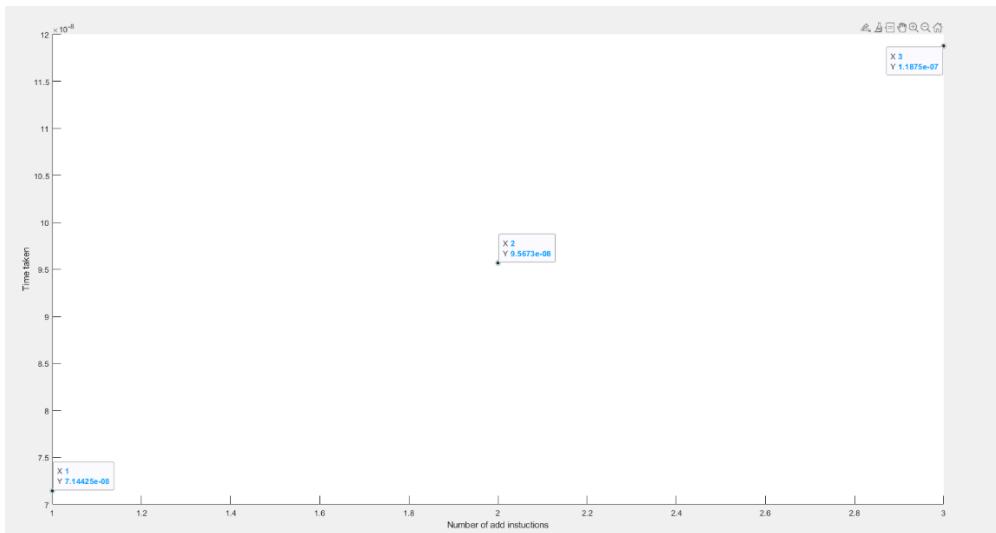


Figure 8: MATLAB plot

Comment/Explanation:

The detail of each condition is shown below:

Address	Time	Num. of add instruction	R ₂ value	Value per second	Time per instruction
0x000000288	20.15s	1	0x10C152A2 281105058 ₍₁₀₎	13950623.23 ₍₁₀₎	71.681*10 ⁻⁹ s
0x000000288	40.35s	1	0x21C6D323 566678307 ₍₁₀₎	14044072.04 ₍₁₀₎	71.204*10 ⁻⁹ s
0x00000028A	20.10s	2	0x0C832865 209921149 ₍₁₀₎	10443838.26 ₍₁₀₎	95.750*10 ⁻⁹ s
0x00000028A	40.09s	2	0x18FEE461 419357793 ₍₁₀₎	10460408.9 ₍₁₀₎	95.596*10 ⁻⁹ s
0x000000292	20.00s	3	0xA094356 168379222 ₍₁₀₎	8418961.1 ₍₁₀₎	1.188*10 ⁻⁷ s
0x000000292	40.15s	3	0x142B0D88	8427542.516 ₍₁₀₎	1.187*10 ⁻⁷ s

From the table, the time average time for one instruction is about 71.4425ns. The average time for two instruction is about 95.673ns, while the average time for three instructions is about 1.1875*10⁻⁷s. These values can be seen on the MATLAB plot.

From the graph, it can be seen that the three points are roughly a straight line. That is also prove that the time for execute instruction is a certain value.

13. Answer to Q12 [4 marks]**Answer:**

23.254s

Explanation:

By taking the results from section 9, the calculation steps are shown below:

$$(95.673n - 71.4425n)/2 = 23.077ns$$

$$(1.1875 \times 10^{-7} - 95.673n)/2 = 24.2305\text{ns}$$

Therefore, the time for each instruction is about

$$(23.077n + 24.2305n)/2 = 23.654\text{ns}.$$

Since add instructions takes one clock to execute, the clock cycle shall be 23.564ns.

14. Answer to Q13 [4 marks]

Answer:

About 2 clock cycles.

Explanation:

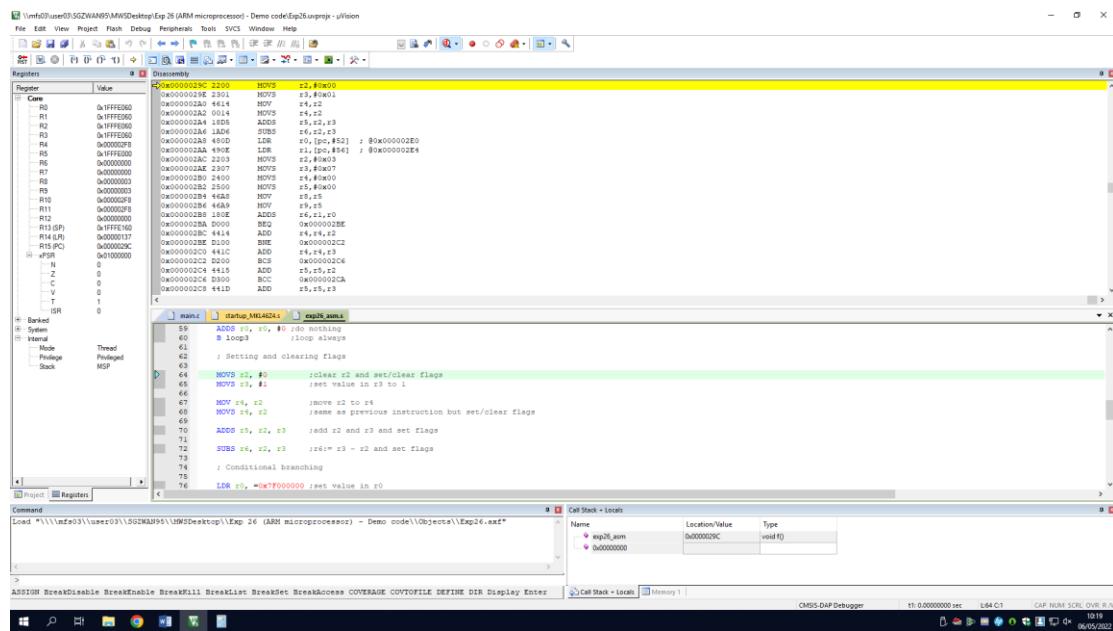
By taking the loop contain one add instruction to calculate the time.

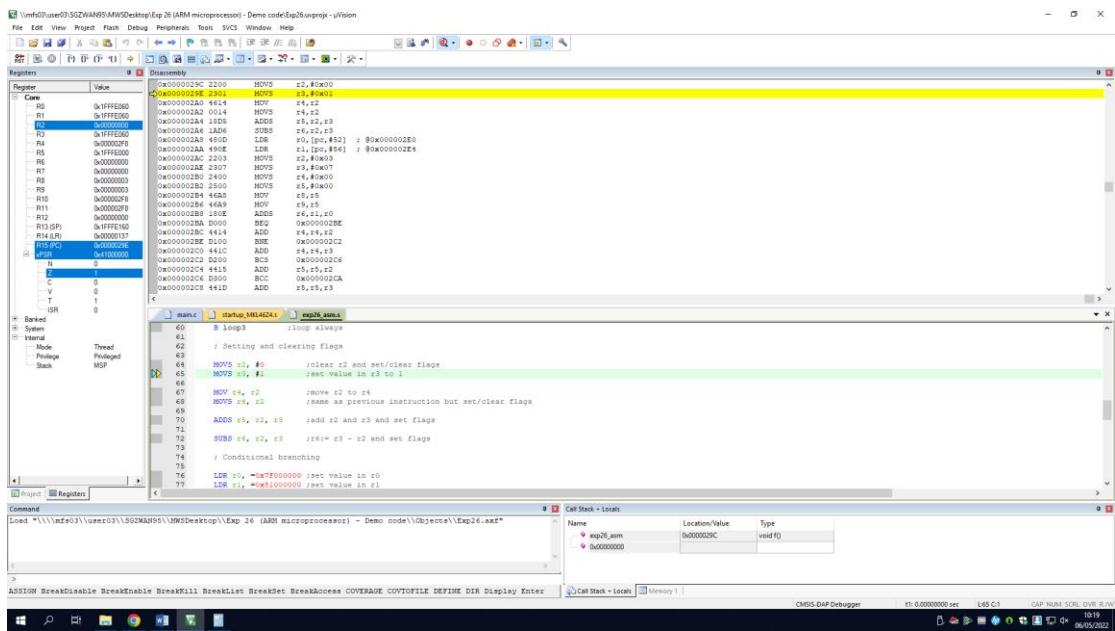
$$(71.4425 - 23.654)/23.654 = 2.02$$

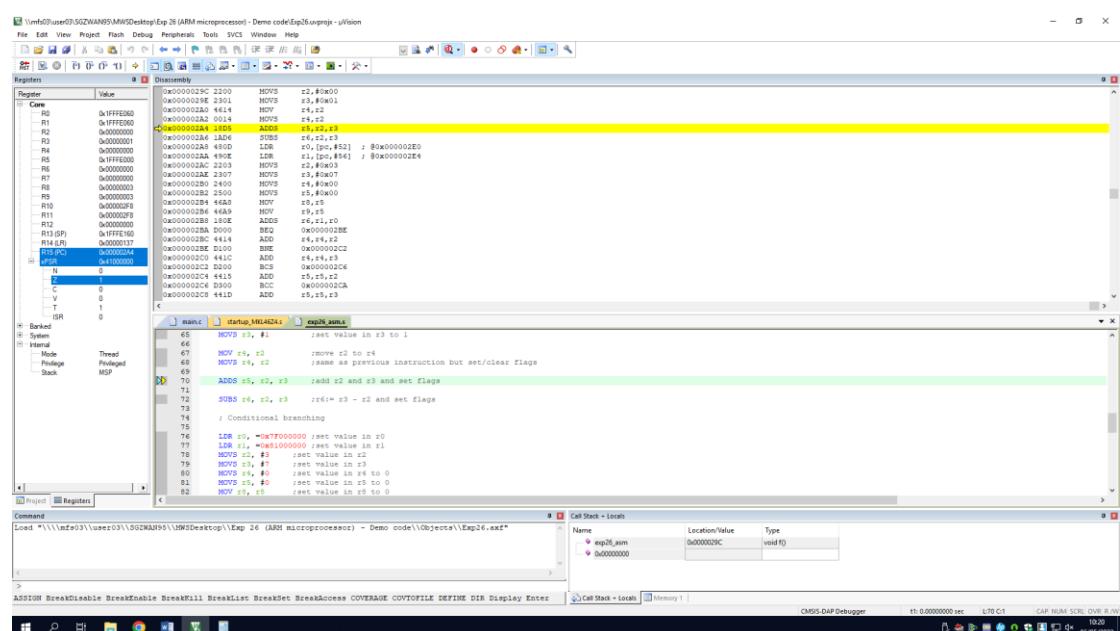
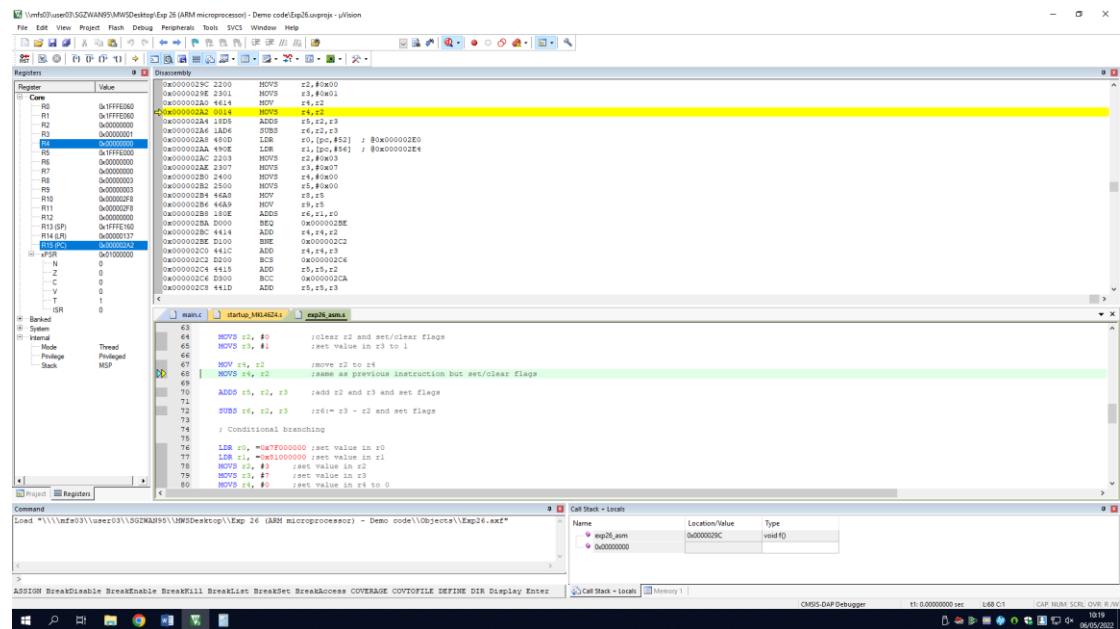
Therefore, the required clock cycle is about 2.

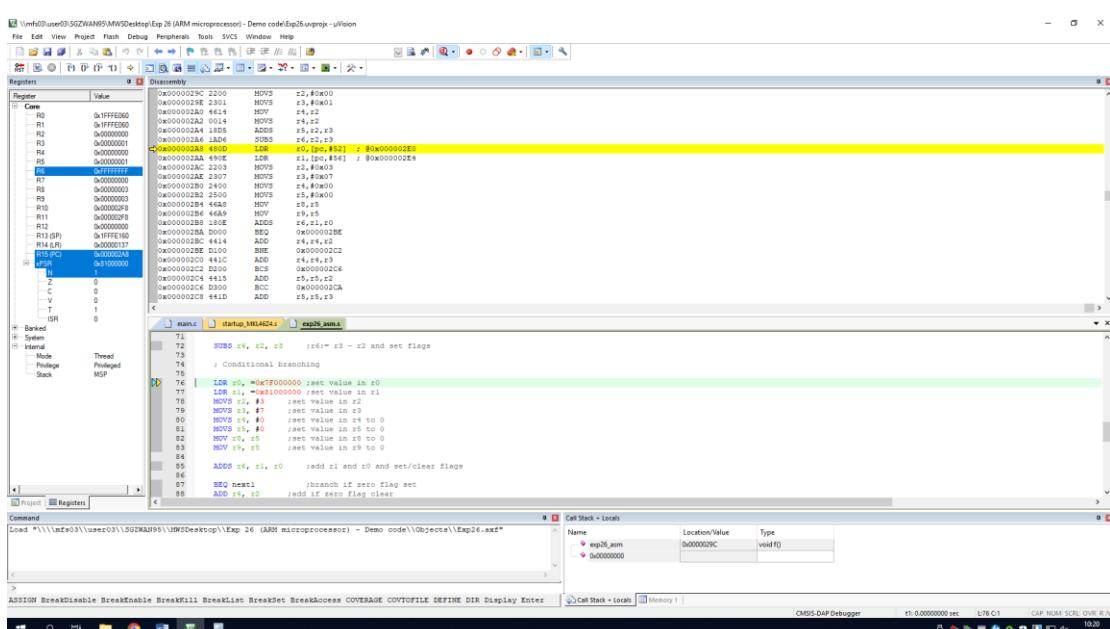
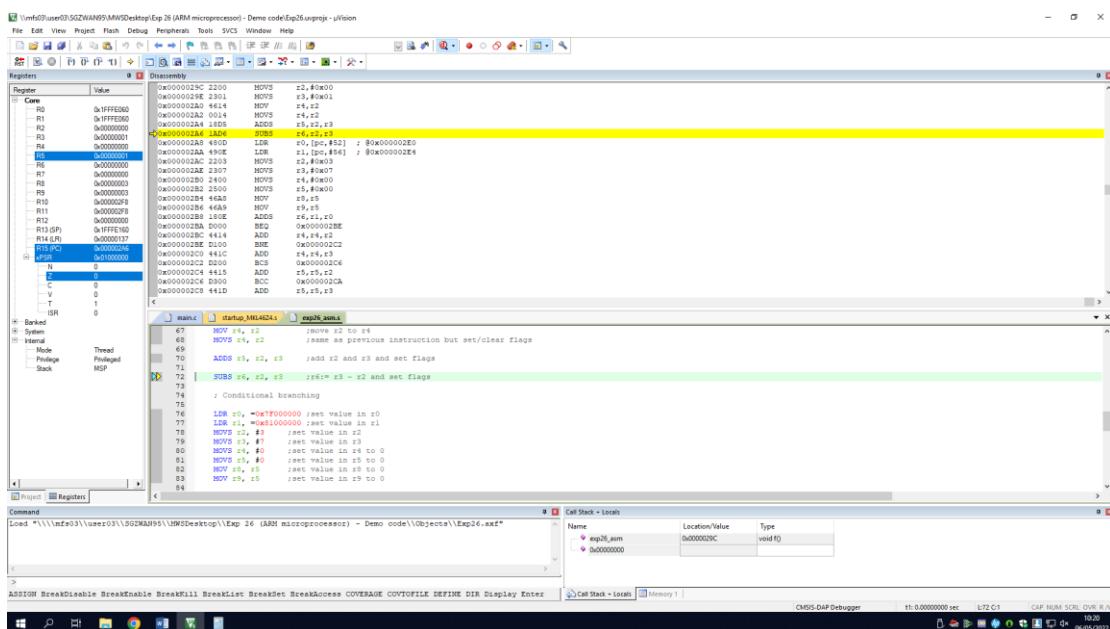
Screenshot of the result of Section 10 [3 marks]

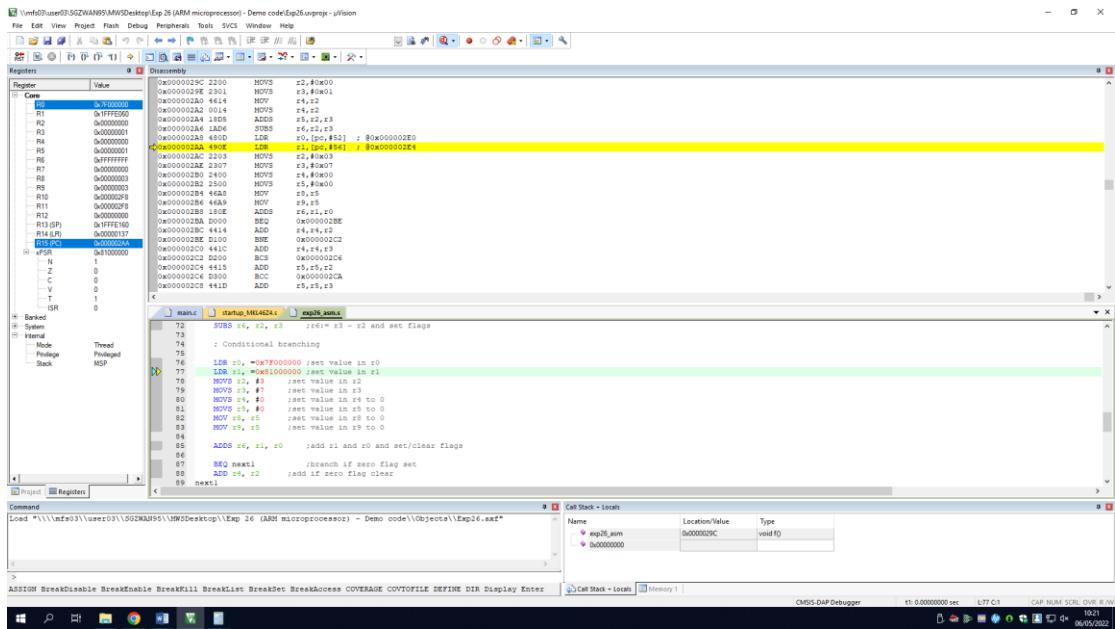
Screenshot (PrintScreen or Screenshot):



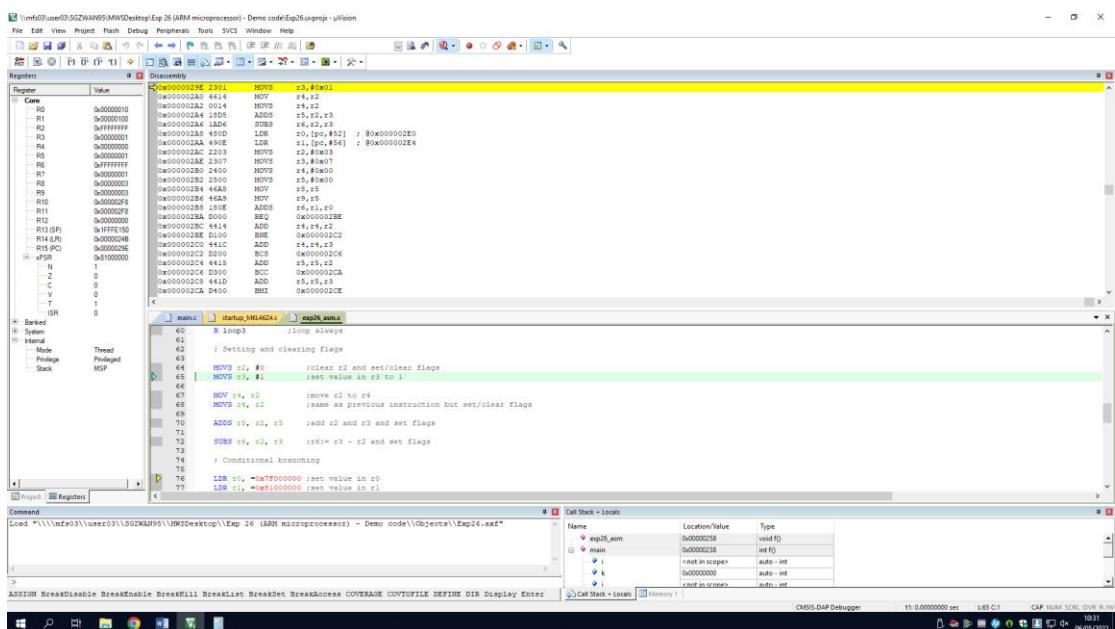


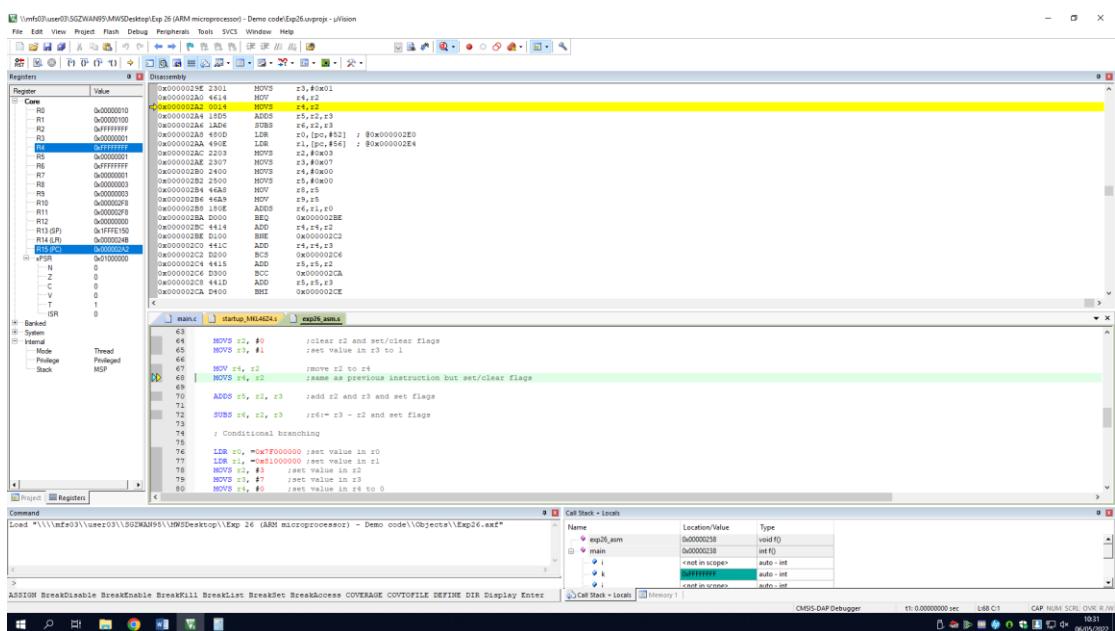
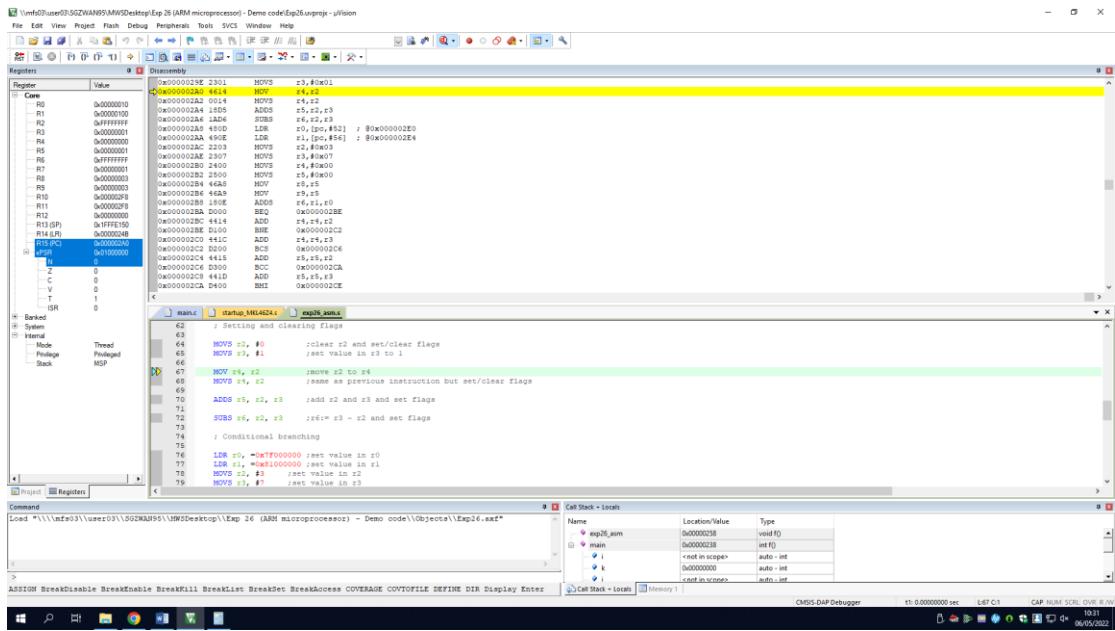


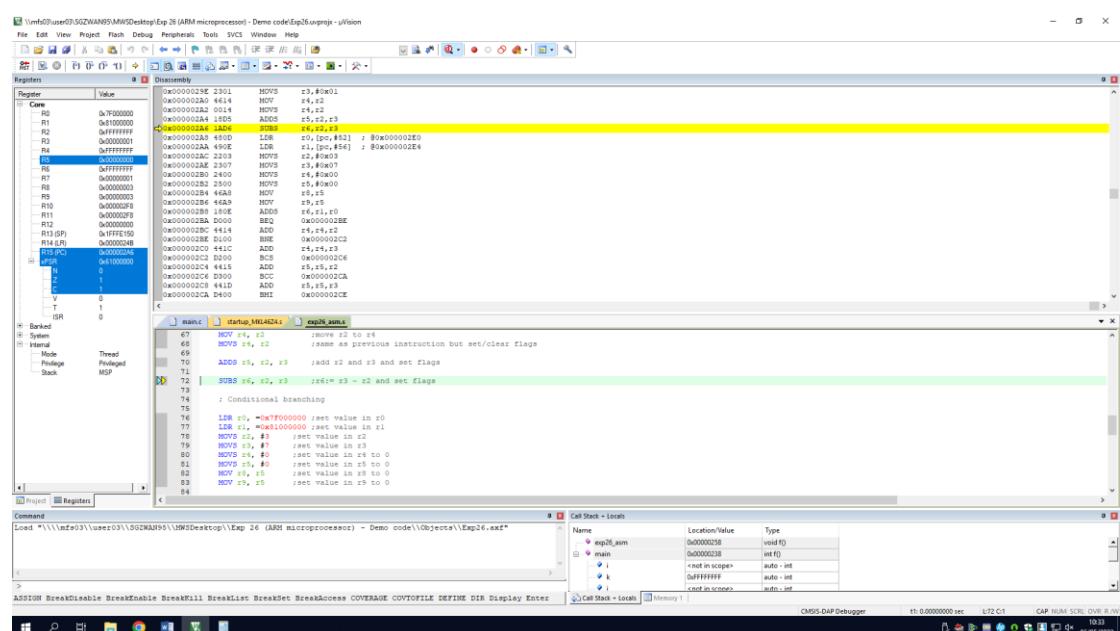
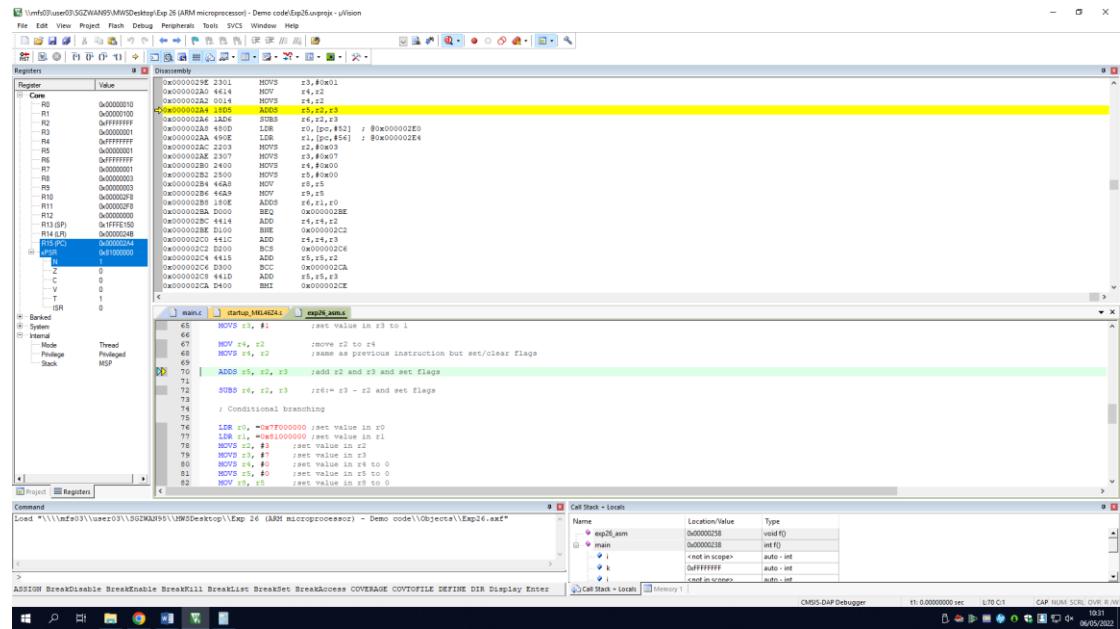


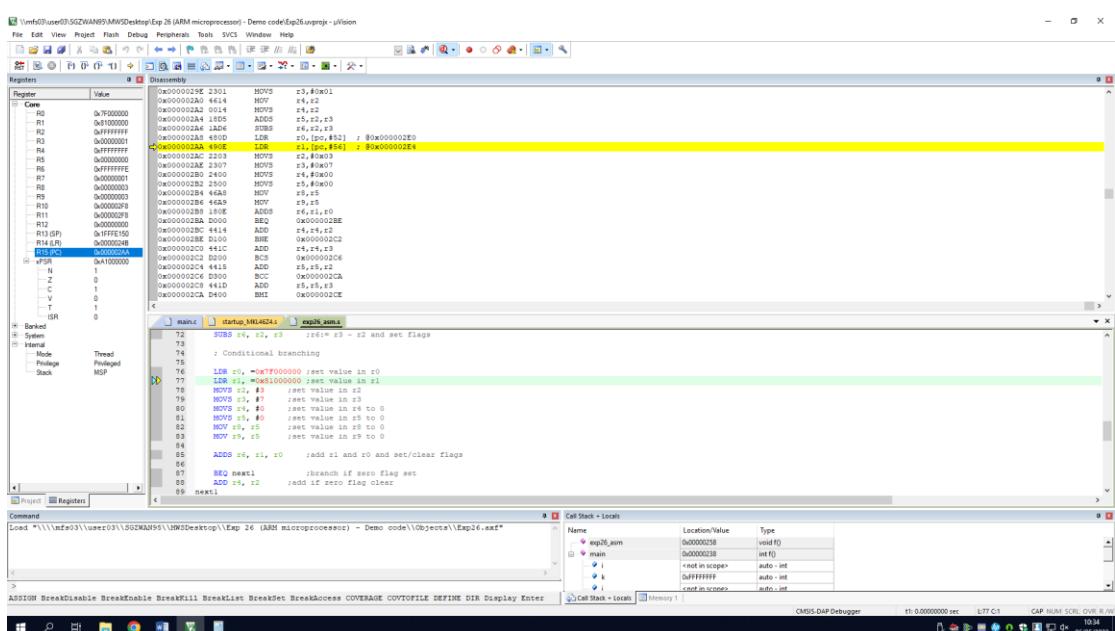
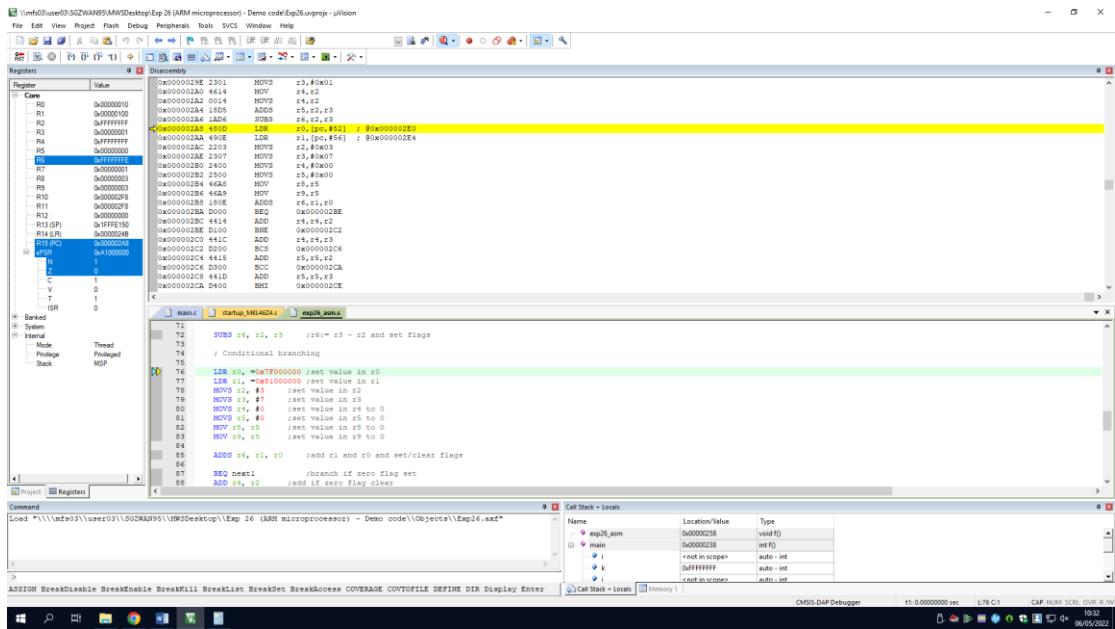


Change value of R₂ to 0xFFFFFFFF

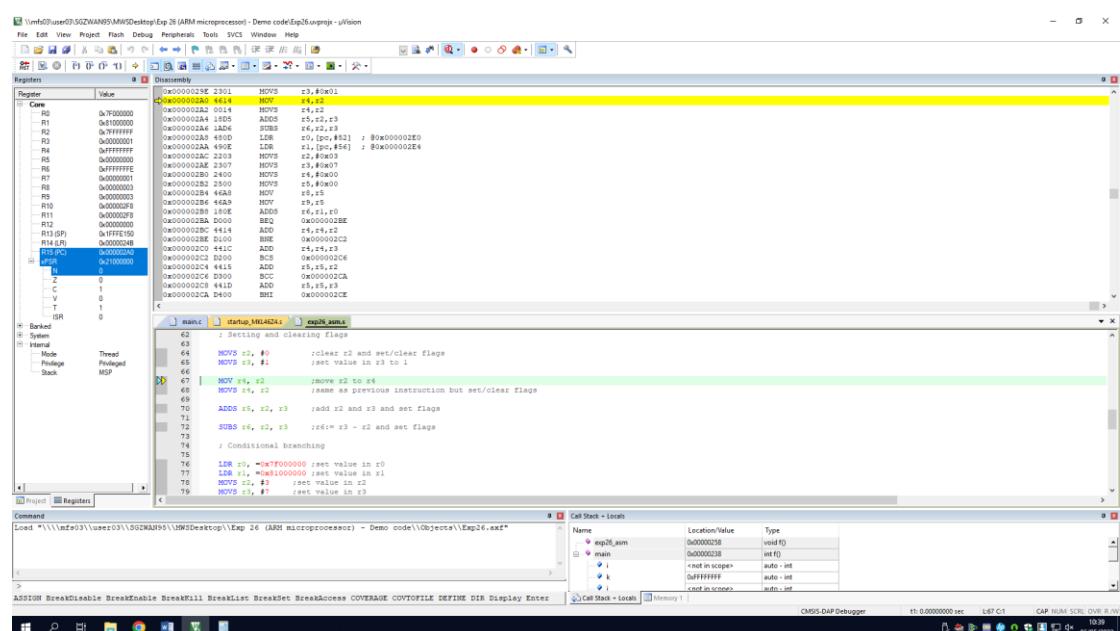
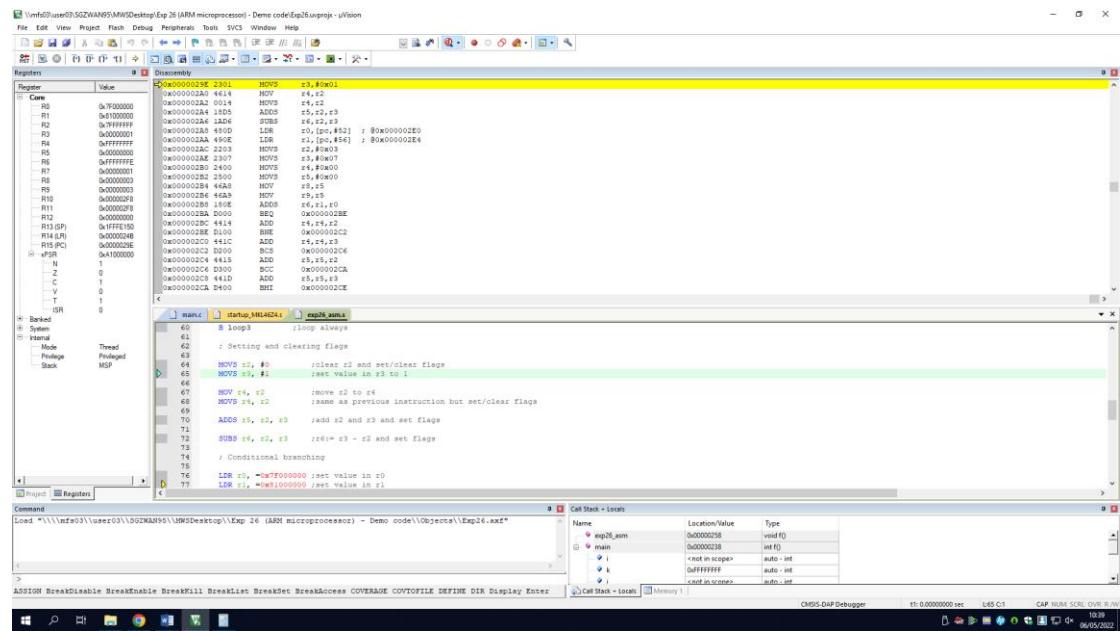


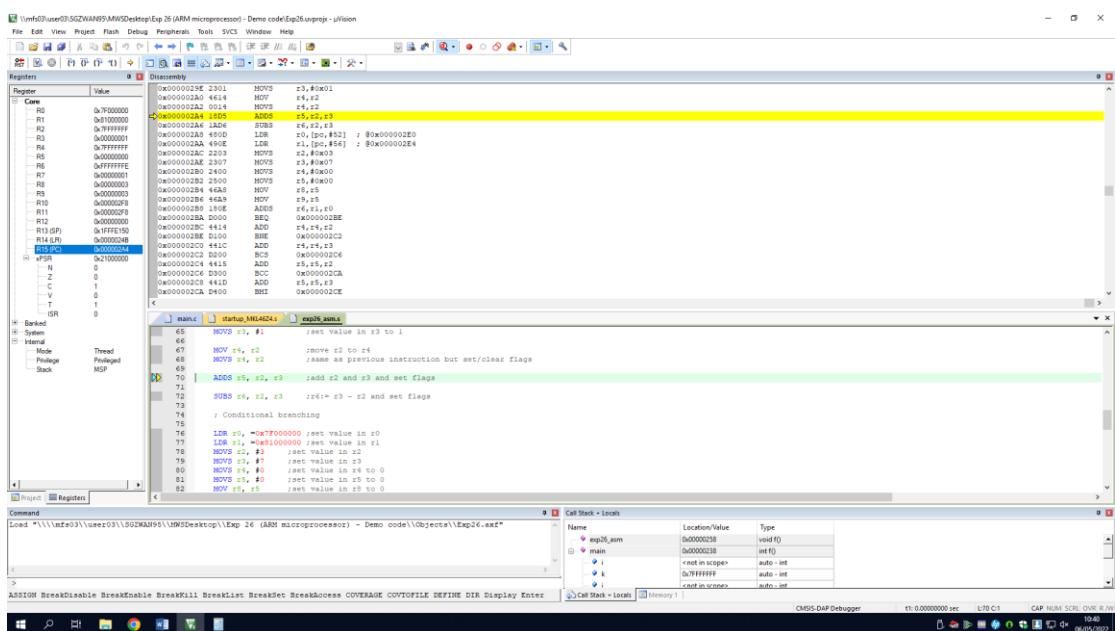
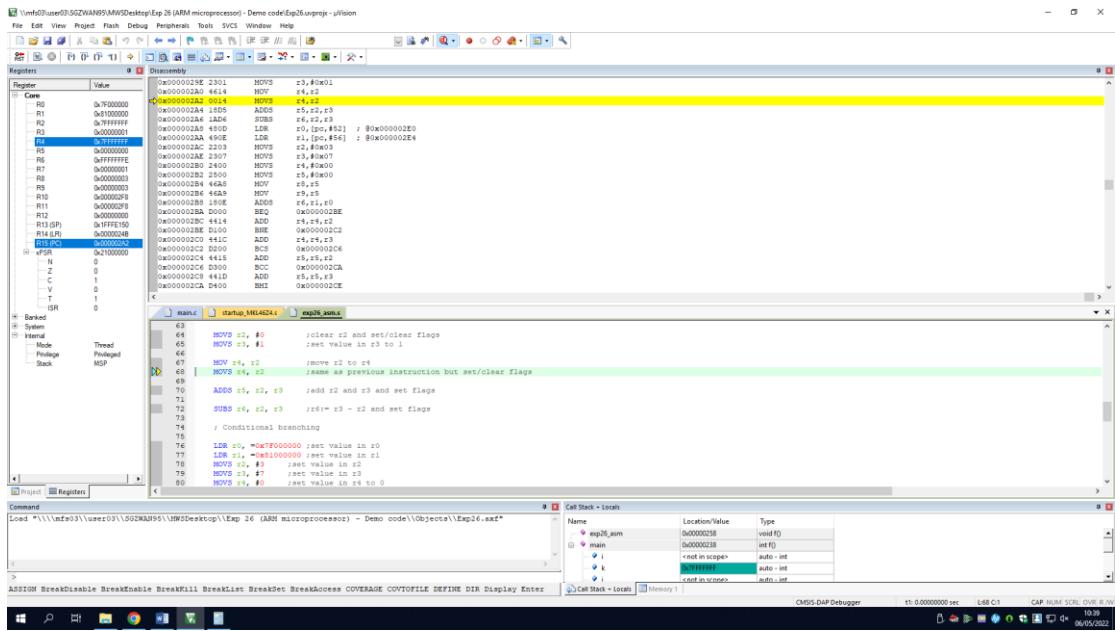






Set R2 0x7FFFFFFF



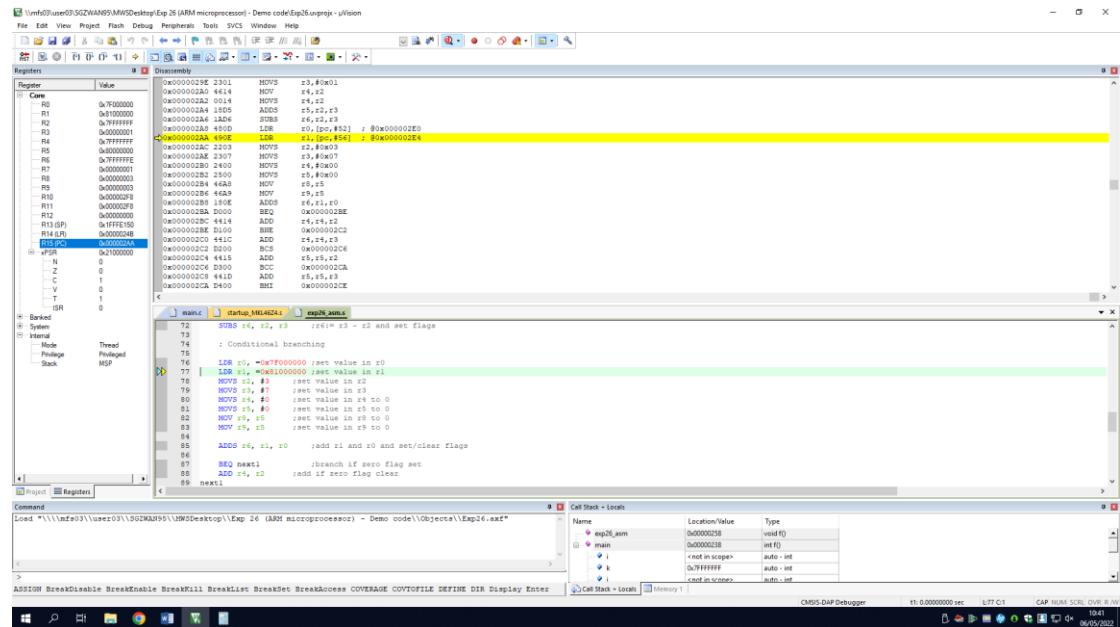


Screenshot of the CMSIS-DAP Debugger interface showing the assembly code for Exp 26 (ARM microprocessor) - Demo code\Exp26\uproj - uVision.

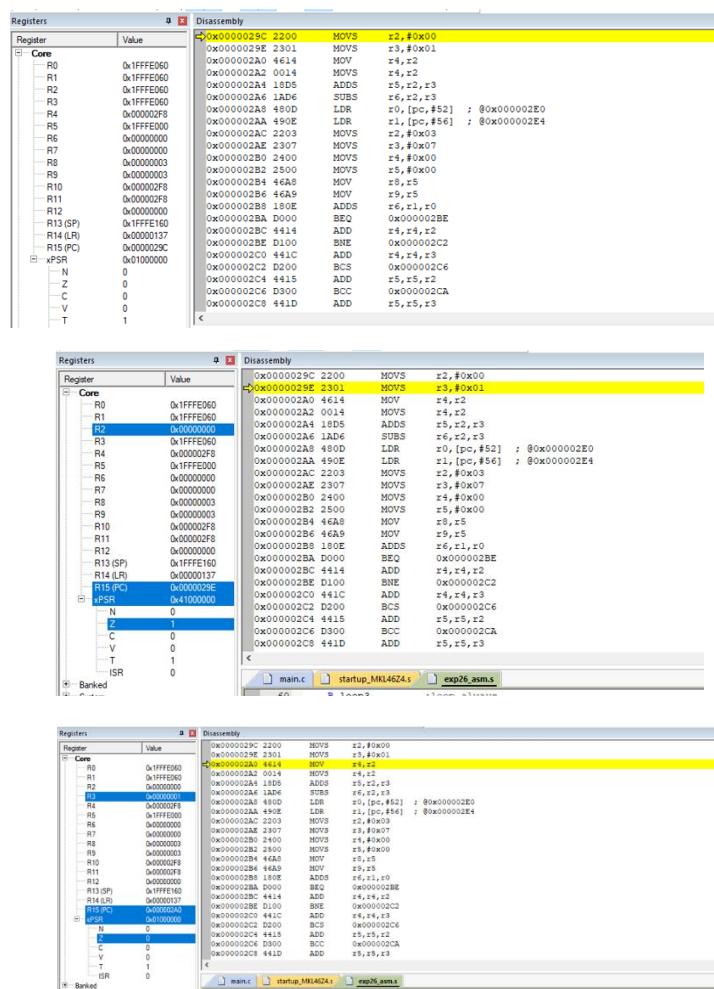
The assembly code is as follows:

```

    .file "main.c"
    .text
    .globl _start
_start:
    .L1:    MOVW r0, #0x0001
    .L2:    MOVS r1, r0
    .L3:    ADDS r2, r0, r1
    .L4:    SUBS r3, r2, r1
    .L5:    BEQ next1, .L6
    .L6:    ADDS r4, r0, r2
    .L7:    SUBS r5, r4, r2
    .L8:    BEQ next2, .L9
    .L9:    ADDS r6, r0, r4
    .L10:   SUBS r7, r6, r4
    .L11:   BEQ next3, .L12
    .L12:   ADDS r8, r0, r6
    .L13:   SUBS r9, r8, r6
    .L14:   BEQ next4, .L15
    .L15:   ADDS r10, r0, r8
    .L16:   SUBS r11, r10, r8
    .L17:   BEQ next5, .L18
    .L18:   ADDS r12, r0, r10
    .L19:   SUBS r13, r12, r10
    .L20:   BEQ next6, .L21
    .L21:   ADDS r14, r0, r12
    .L22:   SUBS r15, r14, r12
    .L23:   BEQ next7, .L24
    .L24:   ADDS r16, r0, r14
    .L25:   SUBS r17, r16, r14
    .L26:   BEQ next8, .L27
    .L27:   ADDS r18, r0, r16
    .L28:   SUBS r19, r18, r16
    .L29:   BEQ next9, .L30
    .L30:   ADDS r20, r0, r16
    .L31:   SUBS r21, r20, r16
    .L32:   BEQ next10, .L33
    .L33:   ADDS r22, r0, r20
    .L34:   SUBS r23, r22, r20
    .L35:   BEQ next11, .L36
    .L36:   ADDS r24, r0, r22
    .L37:   SUBS r25, r24, r22
    .L38:   BEQ next12, .L39
    .L39:   ADDS r26, r0, r24
    .L40:   SUBS r27, r26, r24
    .L41:   BEQ next13, .L42
    .L42:   ADDS r28, r0, r26
    .L43:   SUBS r29, r28, r26
    .L44:   BEQ next14, .L45
    .L45:   ADDS r30, r0, r26
    .L46:   SUBS r31, r30, r26
    .L47:   BEQ next15, .L48
    .L48:   ADDS r32, r0, r26
    .L49:   SUBS r33, r32, r26
    .L50:   BEQ next16, .L51
    .L51:   ADDS r34, r0, r26
    .L52:   SUBS r35, r34, r26
    .L53:   BEQ next17, .L54
    .L54:   ADDS r36, r0, r26
    .L55:   SUBS r37, r36, r26
    .L56:   BEQ next18, .L57
    .L57:   ADDS r38, r0, r26
    .L58:   SUBS r39, r38, r26
    .L59:   BEQ next19, .L60
    .L60:   ADDS r40, r0, r26
    .L61:   SUBS r41, r40, r26
    .L62:   BEQ next20, .L63
    .L63:   ADDS r42, r0, r26
    .L64:   SUBS r43, r42, r26
    .L65:   BEQ next21, .L66
    .L66:   ADDS r44, r0, r26
    .L67:   SUBS r45, r44, r26
    .L68:   BEQ next22, .L69
    .L69:   ADDS r46, r0, r26
    .L70:   SUBS r47, r46, r26
    .L71:   BEQ next23, .L72
    .L72:   ADDS r48, r0, r26
    .L73:   SUBS r49, r48, r26
    .L74:   BEQ next24, .L75
    .L75:   ADDS r50, r0, r26
    .L76:   SUBS r51, r50, r26
    .L77:   BEQ next25, .L78
    .L78:   ADDS r52, r0, r26
    .L79:   SUBS r53, r52, r26
    .L80:   BEQ next26, .L81
    .L81:   ADDS r54, r0, r26
    .L82:   SUBS r55, r54, r26
    .L83:   BEQ next27, .L84
    .L84:   ADDS r56, r0, r26
    .L85:   SUBS r57, r56, r26
    .L86:   BEQ next28, .L87
    .L87:   ADDS r58, r0, r26
    .L88:   SUBS r59, r58, r26
    .L89:   BEQ next29, .L90
    .L90:   ADDS r60, r0, r26
    .L91:   SUBS r61, r60, r26
    .L92:   BEQ next30, .L93
    .L93:   ADDS r62, r0, r26
    .L94:   SUBS r63, r62, r26
    .L95:   BEQ next31, .L96
    .L96:   ADDS r64, r0, r26
    .L97:   SUBS r65, r64, r26
    .L98:   BEQ next32, .L99
    .L99:   ADDS r66, r0, r26
    .L100:  SUBS r67, r66, r26
    .L101:  BEQ next33, .L102
    .L102:  ADDS r68, r0, r26
    .L103:  SUBS r69, r68, r26
    .L104:  BEQ next34, .L105
    .L105:  ADDS r70, r0, r26
    .L106:  SUBS r71, r70, r26
    .L107:  BEQ next35, .L108
    .L108:  ADDS r72, r0, r26
    .L109:  SUBS r73, r72, r26
    .L110:  BEQ next36, .L111
    .L111:  ADDS r74, r0, r26
    .L112:  SUBS r75, r74, r26
    .L113:  BEQ next37, .L114
    .L114:  ADDS r76, r0, r26
    .L115:  SUBS r77, r76, r26
    .L116:  BEQ next38, .L117
    .L117:  ADDS r78, r0, r26
    .L118:  SUBS r79, r78, r26
    .L119:  BEQ next39, .L120
    .L120:  ADDS r80, r0, r26
    .L121:  SUBS r81, r80, r26
    .L122:  BEQ next40, .L123
    .L123:  ADDS r82, r0, r26
    .L124:  SUBS r83, r82, r26
    .L125:  BEQ next41, .L126
    .L126:  ADDS r84, r0, r26
    .L127:  SUBS r85, r84, r26
    .L128:  BEQ next42, .L129
    .L129:  ADDS r86, r0, r26
    .L130:  SUBS r87, r86, r26
    .L131:  BEQ next43, .L132
    .L132:  ADDS r88, r0, r26
    .L133:  SUBS r89, r88, r26
    .L134:  BEQ next44, .L135
    .L135:  ADDS r90, r0, r26
    .L136:  SUBS r91, r90, r26
    .L137:  BEQ next45, .L138
    .L138:  ADDS r92, r0, r26
    .L139:  SUBS r93, r92, r26
    .L140:  BEQ next46, .L141
    .L141:  ADDS r94, r0, r26
    .L142:  SUBS r95, r94, r26
    .L143:  BEQ next47, .L144
    .L144:  ADDS r96, r0, r26
    .L145:  SUBS r97, r96, r26
    .L146:  BEQ next48, .L147
    .L147:  ADDS r98, r0, r26
    .L148:  SUBS r99, r98, r26
    .L149:  BEQ next49, .L150
    .L150:  ADDS r100, r0, r26
    .L151:  SUBS r101, r100, r26
    .L152:  BEQ next50, .L153
    .L153:  ADDS r102, r0, r26
    .L154:  SUBS r103, r102, r26
    .L155:  BEQ next51, .L156
    .L156:  ADDS r104, r0, r26
    .L157:  SUBS r105, r104, r26
    .L158:  BEQ next52, .L159
    .L159:  ADDS r106, r0, r26
    .L160:  SUBS r107, r106, r26
    .L161:  BEQ next53, .L162
    .L162:  ADDS r108, r0, r26
    .L163:  SUBS r109, r108, r26
    .L164:  BEQ next54, .L165
    .L165:  ADDS r110, r0, r26
    .L166:  SUBS r111, r110, r26
    .L167:  BEQ next55, .L168
    .L168:  ADDS r112, r0, r26
    .L169:  SUBS r113, r112, r26
    .L170:  BEQ next56, .L171
    .L171:  ADDS r114, r0, r26
    .L172:  SUBS r115, r114, r26
    .L173:  BEQ next57, .L174
    .L174:  ADDS r116, r0, r26
    .L175:  SUBS r117, r116, r26
    .L176:  BEQ next58, .L177
    .L177:  ADDS r118, r0, r26
    .L178:  SUBS r119, r118, r26
    .L179:  BEQ next59, .L180
    .L180:  ADDS r120, r0, r26
    .L181:  SUBS r121, r120, r26
    .L182:  BEQ next60, .L183
    .L183:  ADDS r122, r0, r26
    .L184:  SUBS r123, r122, r26
    .L185:  BEQ next61, .L186
    .L186:  ADDS r124, r0, r26
    .L187:  SUBS r125, r124, r26
    .L188:  BEQ next62, .L189
    .L189:  ADDS r126, r0, r26
    .L190:  SUBS r127, r126, r26
    .L191:  BEQ next63, .L192
    .L192:  ADDS r128, r0, r26
    .L193:  SUBS r129, r128, r26
    .L194:  BEQ next64, .L195
    .L195:  ADDS r120, r0, r26
    .L196:  SUBS r121, r120, r26
    .L197:  BEQ next65, .L198
    .L198:  ADDS r122, r0, r26
    .L199:  SUBS r123, r122, r26
    .L200:  BEQ next66, .L201
    .L201:  ADDS r124, r0, r26
    .L202:  SUBS r125, r124, r26
    .L203:  BEQ next67, .L204
    .L204:  ADDS r126, r0, r26
    .L205:  SUBS r127, r126, r26
    .L206:  BEQ next68, .L207
    .L207:  ADDS r128, r0, r26
    .L208:  SUBS r129, r128, r26
    .L209:  BEQ next69, .L210
    .L210:  ADDS r120, r0, r26
    .L211:  SUBS r121, r120, r26
    .L212:  BEQ next70, .L213
    .L213:  ADDS r122, r0, r26
    .L214:  SUBS r123, r122, r26
    .L215:  BEQ next71, .L216
    .L216:  ADDS r124, r0, r26
    .L217:  SUBS r125, r124, r26
    .L218:  BEQ next72, .L219
    .L219:  ADDS r126, r0, r26
    .L220:  SUBS r127, r126, r26
    .L221:  BEQ next73, .L222
    .L222:  ADDS r128, r0, r26
    .L223:  SUBS r129, r128, r26
    .L224:  BEQ next74, .L225
    .L225:  ADDS r120, r0, r26
    .L226:  SUBS r121, r120, r26
    .L227:  BEQ next75, .L228
    .L228:  ADDS r122, r0, r26
    .L229:  SUBS r123, r122, r26
    .L230:  BEQ next76, .L231
    .L231:  ADDS r124, r0, r26
    .L232:  SUBS r125, r124, r26
    .L233:  BEQ next77, .L234
    .L234:  ADDS r126, r0, r26
    .L235:  SUBS r127, r126, r26
    .L236:  BEQ next78, .L237
    .L237:  ADDS r128, r0, r26
    .L238:  SUBS r129, r128, r26
    .L239:  BEQ next79, .L240
    .L240:  ADDS r120, r0, r26
    .L241:  SUBS r121, r120, r26
    .L242:  BEQ next80, .L243
    .L243:  ADDS r122, r0, r26
    .L244:  SUBS r123, r122, r26
    .L245:  BEQ next81, .L246
    .L246:  ADDS r124, r0, r26
    .L247:  SUBS r125, r124, r26
    .L248:  BEQ next82, .L249
    .L249:  ADDS r126, r0, r26
    .L250:  SUBS r127, r126, r26
    .L251:  BEQ next83, .L252
    .L252:  ADDS r128, r0, r26
    .L253:  SUBS r129, r128, r26
    .L254:  BEQ next84, .L255
    .L255:  ADDS r120, r0, r26
    .L256:  SUBS r121, r120, r26
    .L257:  BEQ next85, .L258
    .L258:  ADDS r122, r0, r26
    .L259:  SUBS r123, r122, r26
    .L260:  BEQ next86, .L261
    .L261:  ADDS r124, r0, r26
    .L262:  SUBS r125, r124, r26
    .L263:  BEQ next87, .L264
    .L264:  ADDS r126, r0, r26
    .L265:  SUBS r127, r126, r26
    .L266:  BEQ next88, .L267
    .L267:  ADDS r128, r0, r26
    .L268:  SUBS r129, r128, r26
    .L269:  BEQ next89, .L270
    .L270:  ADDS r120, r0, r26
    .L271:  SUBS r121, r120, r26
    .L272:  BEQ next90, .L273
    .L273:  ADDS r122, r0, r26
    .L274:  SUBS r123, r122, r26
    .L275:  BEQ next91, .L276
    .L276:  ADDS r124, r0, r26
    .L277:  SUBS r125, r124, r26
    .L278:  BEQ next92, .L279
    .L279:  ADDS r126, r0, r26
    .L280:  SUBS r127, r126, r26
    .L281:  BEQ next93, .L282
    .L282:  ADDS r128, r0, r26
    .L283:  SUBS r129, r128, r26
    .L284:  BEQ next94, .L285
    .L285:  ADDS r120, r0, r26
    .L286:  SUBS r121, r120, r26
    .L287:  BEQ next95, .L288
    .L288:  ADDS r122, r0, r26
    .L289:  SUBS r123, r122, r26
    .L290:  BEQ next96, .L291
    .L291:  ADDS r124, r0, r26
    .L292:  SUBS r125, r124, r26
    .L293:  BEQ next97, .L294
    .L294:  ADDS r126, r0, r26
    .L295:  SUBS r127, r126, r26
    .L296:  BEQ next98, .L297
    .L297:  ADDS r128, r0, r26
    .L298:  SUBS r129, r128, r26
    .L299:  BEQ next99, .L300
    .L300:  ADDS r120, r0, r26
    .L301:  SUBS r121, r120, r26
    .L302:  BEQ next100, .L303
    .L303:  ADDS r122, r0, r26
    .L304:  SUBS r123, r122, r26
    .L305:  BEQ next101, .L306
    .L306:  ADDS r124, r0, r26
    .L307:  SUBS r125, r124, r26
    .L308:  BEQ next102, .L309
    .L309:  ADDS r126, r0, r26
    .L310:  SUBS r127, r126, r26
    .L311:  BEQ next103, .L312
    .L312:  ADDS r128, r0, r26
    .L313:  SUBS r129, r128, r26
    .L314:  BEQ next104, .L315
    .L315:  ADDS r120, r0, r26
    .L316:  SUBS r121, r120, r26
    .L317:  BEQ next105, .L318
    .L318:  ADDS r122, r0, r26
    .L319:  SUBS r123, r122, r26
    .L320:  BEQ next106, .L321
    .L321:  ADDS r124, r0, r26
    .L322:  SUBS r125, r124, r26
    .L323:  BEQ next107, .L324
    .L324:  ADDS r126, r0, r26
    .L325:  SUBS r127, r126, r26
    .L326:  BEQ next108, .L327
    .L327:  ADDS r128, r0, r26
    .L328:  SUBS r129, r128, r26
    .L329:  BEQ next109, .L330
    .L330:  ADDS r120, r0, r26
    .L331:  SUBS r121, r120, r26
    .L332:  BEQ next110, .L333
    .L333:  ADDS r122, r0, r26
    .L334:  SUBS r123, r122, r26
    .L335:  BEQ next111, .L336
    .L336:  ADDS r124, r0, r26
    .L337:  SUBS r125, r124, r26
    .L338:  BEQ next112, .L339
    .L339:  ADDS r126, r0, r26
    .L340:  SUBS r127, r126, r26
    .L341:  BEQ next113, .L342
    .L342:  ADDS r128, r0, r26
    .L343:  SUBS r129, r128, r26
    .L344:  BEQ next114, .L345
    .L345:  ADDS r120, r0, r26
    .L346:  SUBS r121, r120, r26
    .L347:  BEQ next115, .L348
    .L348:  ADDS r122, r0, r26
    .L349:  SUBS r123, r122, r26
    .L350:  BEQ next116, .L351
    .L351:  ADDS r124, r0, r26
    .L352:  SUBS r125, r124, r26
    .L353:  BEQ next117, .L354
    .L354:  ADDS r126, r0, r26
    .L355:  SUBS r127, r126, r26
    .L356:  BEQ next118, .L357
    .L357:  ADDS r128, r0, r26
    .L358:  SUBS r129, r128, r26
    .L359:  BEQ next119, .L360
    .L360:  ADDS r120, r0, r26
    .L361:  SUBS r121, r120, r26
    .L362:  BEQ next120, .L363
    .L363:  ADDS r122, r0, r26
    .L364:  SUBS r123, r122, r26
    .L365:  BEQ next121, .L366
    .L366:  ADDS r124, r0, r26
    .L367:  SUBS r125, r124, r26
    .L368:  BEQ next122, .L369
    .L369:  ADDS r126, r0, r26
    .L370:  SUBS r127, r126, r26
    .L371:  BEQ next123, .L372
    .L372:  ADDS r128, r0, r26
    .L373:  SUBS r129, r128, r26
    .L374:  BEQ next124, .L375
    .L375:  ADDS r120, r0, r26
    .L376:  SUBS r121, r120, r26
    .L377:  BEQ next125, .L378
    .L378:  ADDS r122, r0, r26
    .L379:  SUBS r123, r122, r26
    .L380:  BEQ next126, .L381
    .L381:  ADDS r124, r0, r26
    .L382:  SUBS r125, r124, r26
    .L383:  BEQ next127, .L384
    .L384:  ADDS r126, r0, r26
    .L385:  SUBS r127, r126, r26
    .L386:  BEQ next128, .L387
    .L387:  ADDS r128, r0, r26
    .L388:  SUBS r129, r128, r26
    .L389:  BEQ next129, .L390
    .L390:  ADDS r120, r0, r26
    .L391:  SUBS r121, r120, r26
    .L392:  BEQ next130, .L393
    .L393:  ADDS r122, r0, r26
    .L394:  SUBS r123, r122, r26
    .L395:  BEQ next131, .L396
    .L396:  ADDS r124, r0, r26
    .L397:  SUBS r125, r124, r26
    .L398:  BEQ next132, .L399
    .L399:  ADDS r126, r0, r26
    .L400:  SUBS r127, r126, r26
    .L401:  BEQ next133, .L402
    .L402:  ADDS r128, r0, r26
    .L403:  SUBS r129, r128, r26
    .L404:  BEQ next134, .L405
    .L405:  ADDS r120, r0, r26
    .L406:  SUBS r121, r120, r26
    .L407:  BEQ next135, .L408
    .L408:  ADDS r122, r0, r26
    .L409:  SUBS r123, r122, r26
    .L410:  BEQ next136, .L411
    .L411:  ADDS r124, r0, r26
    .L412:  SUBS r125, r124, r26
    .L413:  BEQ next137, .L414
    .L414:  ADDS r126, r0, r26
    .L415:  SUBS r127, r126, r26
    .L416:  BEQ next138, .L417
    .L417:  ADDS r128, r0, r26
    .L418:  SUBS r129, r128, r26
    .L419:  BEQ next139, .L420
    .L420:  ADDS r120, r0, r26
    .L421:  SUBS r121, r120, r26
    .L422:  BEQ next140, .L423
    .L423:  ADDS r122, r0, r26
    .L424:  SUBS r123, r122, r26
    .L425:  BEQ next141, .L426
    .L426:  ADDS r124, r0, r26
    .L427:  SUBS r125, r124, r26
    .L428:  BEQ next142, .L429
    .L429:  ADDS r126, r0, r26
    .L430:  SUBS r127, r126, r26
    .L431:  BEQ next143, .L432
    .L432:  ADDS r128, r0, r26
    .L433:  SUBS r129, r128, r26
    .L434:  BEQ next144, .L435
    .L435:  ADDS r120, r0, r26
    .L436:  SUBS r121, r120, r26
    .L437:  BEQ next145, .L438
    .L438:  ADDS r122, r0, r26
    .L439:  SUBS r123, r122, r26
    .L440:  BEQ next146, .L441
    .L441:  ADDS r124, r0, r26
    .L442:  SUBS r125, r124, r26
    .L443:  BEQ next147, .L444
    .L444:  ADDS r126, r0, r26
    .L445:  SUBS r127, r126, r26
    .L446:  BEQ next148, .L447
    .L447:  ADDS r128, r0, r26
    .L448:  SUBS r129, r128, r26
    .L449:  BEQ next149, .L450
    .L450:  ADDS r120, r0, r26
    .L451:  SUBS r121, r120, r26
    .L452:  BEQ next150, .L453
    .L453:  ADDS r122, r0, r26
    .L454:  SUBS r123, r122, r26
    .L455:  BEQ next151, .L456
    .L456:  ADDS r124, r0, r26
    .L457:  SUBS r125, r124, r26
    .L458:  BEQ next152, .L459
    .L459:  ADDS r126, r0, r26
    .L460:  SUBS r127, r126, r26
    .L461:  BEQ next153, .L462
    .L462:  ADDS r128, r0, r26
    .L463:  SUBS r129, r128, r26
    .L464:  BEQ next154, .L465
    .L465:  ADDS r120, r0, r26
    .L466:  SUBS r121, r120, r26
    .L467:  BEQ next155, .L468
    .L468:  ADDS r122, r0, r26
    .L469:  SUBS r123, r122, r26
    .L470:  BEQ next156, .L471
    .L471:  ADDS r124, r0, r26
    .L472:  SUBS r125, r124, r26
    .L473:  BEQ next157, .L474
    .L474:  ADDS r126, r0, r26
    .L475:  SUBS r127, r126, r26
    .L476:  BEQ next158, .L477
    .L477:  ADDS r128, r0, r26
    .L478:  SUBS r129, r128, r26
    .L479:  BEQ next159, .L480
    .L480:  ADDS r120, r0, r26
    .L481:  SUBS r121, r120, r26
    .L482:  BEQ next160, .L483
    .L483:  ADDS r122, r0, r26
    .L484:  SUBS r123, r122, r26
    .L485:  BEQ next161, .L486
    .L486:  ADDS r124, r0, r26
    .L487:  SUBS r125, r124, r26
    .L488:  BEQ next162, .L489
    .L489:  ADDS r126, r0, r26
    .L490:  SUBS r127, r126, r26
    .L491:  BEQ next163, .L492
    .L492:  ADDS r128, r0, r26
    .L493:  SUBS r129, r128, r26
    .L494:  BEQ next164, .L495
    .L495:  ADDS r120, r0, r26
    .L496:  SUBS r121, r120, r26
    .L497:  BEQ next165, .L498
    .L498:  ADDS r122, r0, r26
    .L499:  SUBS r123, r122, r26
    .L500:  BEQ next166, .L501
    .L501:  ADDS r124, r0, r26
    .L502:  SUBS r125, r124, r26
    .L503:  BEQ next167, .L504
    .L504:  ADDS r126, r0, r26
    .L505:  SUBS r127, r126, r26
    .L506:  BEQ next168, .L507
    .L507:  ADDS r128, r0, r26
    .L508:  SUBS r129, r128, r26
    .L509:  BEQ next169, .L510
    .L510:  ADDS r120, r0, r26
    .L511:  SUBS r121, r120, r26
    .L512:  BEQ next170, .L513
    .L513:  ADDS r122, r0, r26
    .L514
```



Screenshot (Snipping tool, Preview or equivalent):



Registers

Register	Value
R0	0xFFFFE600
R1	0xFFFFE600
R2	0x00000000
R3	0x00000001
R4	0x00000000
R5	0x00000001
R6	0xFFFFE600
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000001
R11	0x00000000
R12	0x00000000
R13 (SP)	0xFFFFE160
R14 (LR)	0x00000137
R15 (PC)	0x000002A4
xPSR	0x10000000
N	0
Z	1
C	0
V	0
T	1
ISR	0

Disassembly

```

0x00000029C 2200 MOV.S r2,$0x00
0x00000029E 2301 MOV.S r3,$0x001
0x0000002A0 4614 NOT r4,r2
0x0000002A1 10D5 ADDS r5,r2,r3
0x0000002A2 0014 MOV.S r4,r2
0x0000002A3 480D LDR r0,[pc,$52] ; @0x0000002E0
0x0000002A4 1AD6 SUBS r6,r2,r3
0x0000002A5 480B LDR r0,[pc,$52] ; @0x0000002E0
0x0000002A6 490E LDR r1,[pc,$56] ; @0x0000002E4
0x0000002A7 2307 MOV.S r3,$0x007
0x0000002A8 2300 MOV.S r4,$0x00
0x0000002A9 4614 NOT r5,r2
0x0000002AA 10D5 ADDS r6,r2,r3
0x0000002AB 0014 MOV.S r4,r2
0x0000002AC 480D LDR r0,[pc,$52] ; @0x0000002E0
0x0000002AD 490E LDR r1,[pc,$56] ; @0x0000002E4
0x0000002AE 2307 MOV.S r3,$0x007
0x0000002AF 2300 MOV.S r4,$0x00
0x0000002B0 4614 NOT r5,r2
0x0000002B1 10D5 ADDS r6,r2,r3
0x0000002B2 0014 MOV.S r4,r2
0x0000002B3 480D LDR r0,[pc,$52] ; @0x0000002E0
0x0000002B4 490E LDR r1,[pc,$56] ; @0x0000002E4
0x0000002B5 2307 MOV.S r3,$0x007
0x0000002B6 2300 MOV.S r4,$0x00
0x0000002B7 4614 NOT r5,r2
0x0000002B8 180E ADDS r6,r1,r0
0x0000002B9 0014 MOV.S r4,r2
0x0000002BA D000 BEQ 0x0000002BE
0x0000002BC 4414 ADD r4,r4,r2
0x0000002BD D100 BNE 0x0000002C2
0x0000002CE 441C ADD r4,r4,r3
0x0000002CF D200 BCS 0x0000002C6
0x0000002D0 4415 ADD r5,r5,r2
0x0000002D1 D300 BCC 0x0000002CA
0x0000002D2 441D ADD r5,r5,r3

```

Registers

Register	Value
R0	0xFFFFE600
R1	0xFFFFE600
R2	0x00000000
R3	0x00000001
R4	0x00000000
R5	0x00000001
R6	0xFFFFFFFF
R7	0x00000000
R8	0x00000000
R9	0x00000003
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0xFFFFE160
R14 (LR)	0x00000137
R15 (PC)	0x000002A4
xPSR	0x10000000
N	1
Z	0
C	0
V	0
T	1
ISR	0

Disassembly

```

0x00000029C 2200 MOV.S r2,$0x00
0x00000029E 2301 MOV.S r3,$0x001
0x0000002A0 4614 MOV.S r4,r2
0x0000002A1 0014 MOV.S r4,r2
0x0000002A2 480D ADDS r5,r2,r3
0x0000002A3 1AD6 SUBS r6,r2,r3
0x0000002A4 480B LDR r0,[pc,$52] ; @0x0000002E0
0x0000002A5 490E LDR r1,[pc,$56] ; @0x0000002E4
0x0000002A6 2307 MOV.S r3,$0x007
0x0000002A7 2300 MOV.S r4,$0x00
0x0000002A8 4614 NOT r5,r2
0x0000002A9 10D5 ADDS r6,r2,r3
0x0000002AA 0014 MOV.S r4,r2
0x0000002AB 480D LDR r0,[pc,$52] ; @0x0000002E0
0x0000002AC 490E LDR r1,[pc,$56] ; @0x0000002E4
0x0000002AD 2307 MOV.S r3,$0x007
0x0000002AE 2300 MOV.S r4,$0x00
0x0000002AF 4614 NOT r5,r2
0x0000002B0 10D5 ADDS r6,r2,r3
0x0000002B1 0014 MOV.S r4,r2
0x0000002B2 480D LDR r0,[pc,$52] ; @0x0000002E0
0x0000002B3 490E LDR r1,[pc,$56] ; @0x0000002E4
0x0000002B4 2307 MOV.S r3,$0x007
0x0000002B5 2300 MOV.S r4,$0x00
0x0000002B6 4614 NOT r5,r2
0x0000002B7 180E ADDS r6,r1,r0
0x0000002B8 0014 MOV.S r4,r2
0x0000002B9 D000 BEQ 0x0000002BE
0x0000002BC 4414 ADD r4,r4,r2
0x0000002BD D100 BNE 0x0000002C2
0x0000002CE 441C ADD r4,r4,r3
0x0000002CF D200 BCS 0x0000002C6
0x0000002D0 4415 ADD r5,r5,r2
0x0000002D1 D300 BCC 0x0000002CA
0x0000002D2 441D ADD r5,r5,r3

```

Registers

Register	Value
R0	0xFFFF0000
R1	0xFFFFE600
R2	0x00000000
R3	0x00000001
R4	0x00000000
R5	0x00000001
R6	0xFFFFFFFF
R7	0x00000000
R8	0x00000000
R9	0x00000003
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0xFFFFE160
R14 (LR)	0x00000137
R15 (PC)	0x000002AA
xPSR	0x81000000
N	1
Z	0
C	0
V	0
T	1
ISR	0

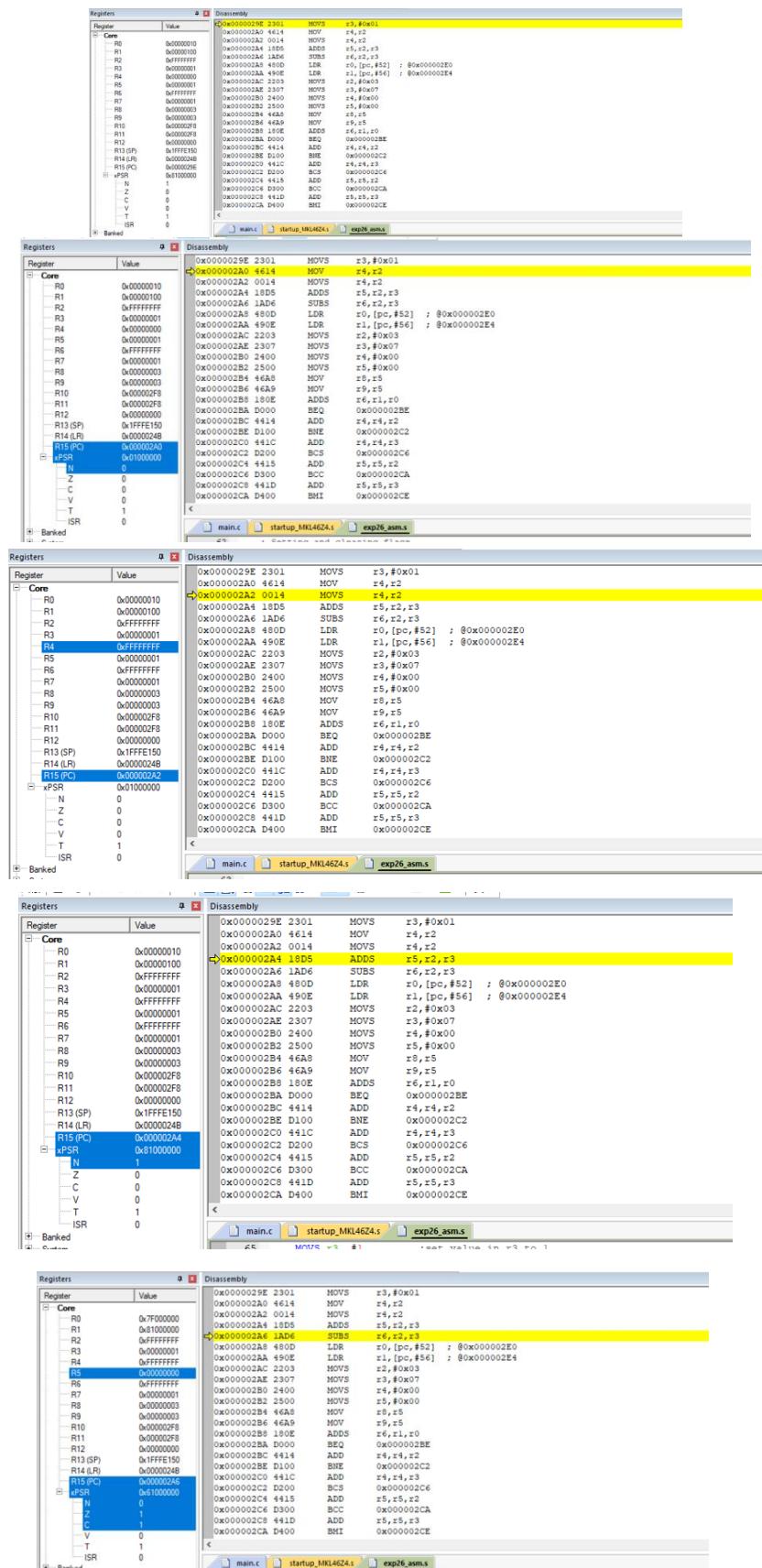
Disassembly

```

0x00000029C 2200 MOV.S r2,$0x00
0x00000029E 2301 MOV.S r3,$0x001
0x0000002A0 4614 MOV.S r4,r2
0x0000002A1 0014 MOV.S r4,r2
0x0000002A2 480D ADDS r5,r2,r3
0x0000002A3 1AD6 SUBS r6,r2,r3
0x0000002A4 480B LDR r0,[pc,$52] ; @0x0000002E0
0x0000002A5 490E LDR r1,[pc,$56] ; @0x0000002E4
0x0000002A6 2307 MOV.S r3,$0x007
0x0000002A7 2300 MOV.S r4,$0x00
0x0000002A8 4614 NOT r5,r2
0x0000002A9 10D5 ADDS r6,r2,r3
0x0000002AA 0014 MOV.S r4,r2
0x0000002AB 480D LDR r0,[pc,$52] ; @0x0000002E0
0x0000002AC 490E LDR r1,[pc,$56] ; @0x0000002E4
0x0000002AD 2307 MOV.S r3,$0x007
0x0000002AE 2300 MOV.S r4,$0x00
0x0000002AF 4614 NOT r5,r2
0x0000002B0 10D5 ADDS r6,r2,r3
0x0000002B1 0014 MOV.S r4,r2
0x0000002B2 480D LDR r0,[pc,$52] ; @0x0000002E0
0x0000002B3 490E LDR r1,[pc,$56] ; @0x0000002E4
0x0000002B4 2307 MOV.S r3,$0x007
0x0000002B5 2300 MOV.S r4,$0x00
0x0000002B6 4614 NOT r5,r2
0x0000002B7 180E ADDS r6,r1,r0
0x0000002B8 0014 MOV.S r4,r2
0x0000002B9 D000 BEQ 0x0000002BE
0x0000002BC 4414 ADD r4,r4,r2
0x0000002BD D100 BNE 0x0000002C2
0x0000002CE 441C ADD r4,r4,r3
0x0000002CF D200 BCS 0x0000002C6
0x0000002D0 4415 ADD r5,r5,r2
0x0000002D1 D300 BCC 0x0000002CA
0x0000002D2 441D ADD r5,r5,r3

```

Change the value to 0xFFFFFFFF



Registers

Register	Value
R0	0x00000010
R1	0x00000100
R2	0xFFFFFFFF
R3	0x00000001
R4	0xFFFFFFFF
R5	0x00000000
R6	0xFFFFFFFF
R7	0x00000001
R8	0x00000003
R9	0x00000003
R10	0x000002F8
R11	0x000002F8
R12	0x00000000
R13 (SP)	0xFFFFE150
R14 (LR)	0x0000024B
R15 (PC)	0x000002A8
xPSR	0xA1000000
N	1
Z	0
C	1
V	0
T	1
ISR	0

Disassembly

```

0x0000029E 2301 MOVS r3,#0x01
0x000002A0 4614 MOV r4,r2
0x000002A2 0014 MOVS r4,r2
0x000002A4 18D5 ADDS r5,r2,r3
0x000002A6 1AD6 SUBS r6,r2,r3
0x000002A8 480D LDR r0,[pc,$52] ; @0x000002E0
0x000002A9 490E LDR r1,[pc,$56] ; @0x000002E4
0x000002A0 2203 MOVS r2,#0x03
0x000002A2 2307 MOVS r3,#0x07
0x000002B0 2400 MOVS r4,#0x00
0x000002B2 2500 MOVS r5,#0x00
0x000002B4 46A8 MOV r8,r5
0x000002B6 46A9 MOV r9,r5
0x000002B8 180E ADDS r6,r1,r0
0x000002B9 D000 BEQ 0x000002BE
0x000002BC 4414 ADD r4,r4,r2
0x000002BD D100 BNE 0x000002C2
0x000002C0 441C ADD r4,r4,r3
0x000002C2 D200 BCS 0x000002C6
0x000002C4 4415 ADD r5,r5,r2
0x000002C6 D300 BCC 0x000002CA
0x000002C8 441D ADD r5,r5,r3
0x000002CA D400 BMI 0x000002CE

```

main.c startup_MKL46Z4.s exp26_asm.s

Registers

Register	Value
R0	0x7F000000
R1	0xB1000000
R2	0xFFFFFFFF
R3	0x00000001
R4	0xFFFFFFFF
R5	0x00000000
R6	0xFFFFFFFF
R7	0x00000001
R8	0x00000003
R9	0x00000003
R10	0x000002F8
R11	0x000002F8
R12	0x00000000
R13 (SP)	0xFFFFE150
R14 (LR)	0x0000024B
R15 (PC)	0x000002A8
xPSR	0xA1000000
N	1
Z	0
C	1
V	0
T	1
ISR	0

Disassembly

```

0x0000029E 2301 MOVS r3,#0x01
0x000002A0 4614 MOV r4,r2
0x000002A2 0014 MOVS r4,r2
0x000002A4 18D5 ADDS r5,r2,r3
0x000002A6 1AD6 SUBS r6,r2,r3
0x000002A8 480D LDR r0,[pc,$52] ; @0x000002E0
0x000002A9 490E LDR r1,[pc,$56] ; @0x000002E4
0x000002A0 2203 MOVS r2,#0x03
0x000002A2 2307 MOVS r3,#0x07
0x000002B0 2400 MOVS r4,#0x00
0x000002B2 2500 MOVS r5,#0x00
0x000002B4 46A8 MOV r8,r5
0x000002B6 46A9 MOV r9,r5
0x000002B8 180E ADDS r6,r1,r0
0x000002B9 D000 BEQ 0x000002BE
0x000002BC 4414 ADD r4,r4,r2
0x000002BD D100 BNE 0x000002C2
0x000002C0 441C ADD r4,r4,r3
0x000002C2 D200 BCS 0x000002C6
0x000002C4 4415 ADD r5,r5,r2
0x000002C6 D300 BCC 0x000002CA
0x000002C8 441D ADD r5,r5,r3
0x000002CA D400 BMI 0x000002CE

```

main.c startup_MKL46Z4.s exp26_asm.s

Set R2 0x7FFFFFFF

Registers

Register	Value
R0	0x7F000000
R1	0xB1000000
R2	0xFFFFFFFF
R3	0x00000001
R4	0xFFFFFFFF
R5	0x00000000
R6	0xFFFFFFFF
R7	0x00000001
R8	0x00000003
R9	0x00000003
R10	0x000002F8
R11	0x000002F8
R12	0x00000000
R13 (SP)	0xFFFFE150
R14 (LR)	0x0000024B
R15 (PC)	0x0000029E
xPSR	0xA1000000
N	1
Z	0
C	1
V	0
T	1
ISR	0

Disassembly

```

0x0000029E 2301 MOVS r3,#0x01
0x000002A0 4614 MOV r4,r2
0x000002A2 0014 MOVS r4,r2
0x000002A4 18D5 ADDS r5,r2,r3
0x000002A6 1AD6 SUBS r6,r2,r3
0x000002A8 480D LDR r0,[pc,$52] ; @0x000002E0
0x000002A9 490E LDR r1,[pc,$56] ; @0x000002E4
0x000002A0 2203 MOVS r2,#0x03
0x000002A2 2307 MOVS r3,#0x07
0x000002B0 2400 MOVS r4,#0x00
0x000002B2 2500 MOVS r5,#0x00
0x000002B4 46A8 MOV r8,r5
0x000002B6 46A9 MOV r9,r5
0x000002B8 180E ADDS r6,r1,r0
0x000002B9 D000 BEQ 0x000002BE
0x000002BC 4414 ADD r4,r4,r2
0x000002BD D100 BNE 0x000002C2
0x000002C0 441C ADD r4,r4,r3
0x000002C2 D200 BCS 0x000002C6
0x000002C4 4415 ADD r5,r5,r2
0x000002C6 D300 BCC 0x000002CA
0x000002C8 441D ADD r5,r5,r3
0x000002CA D400 BMI 0x000002CE

```

Registers

Register	Value
R0	0x7F000000
R1	0xB1000000
R2	0xFFFFFFFF
R3	0x00000001
R4	0xFFFFFFFF
R5	0x00000000
R6	0xFFFFFFFF
R7	0x00000001
R8	0x00000003
R9	0x00000003
R10	0x000002F8
R11	0x000002F8
R12	0x00000000
R13 (SP)	0xFFFFE150
R14 (LR)	0x0000024B
R15 (PC)	0x000002A0
xPSR	0x21000000
N	0
Z	0
C	1
V	0
T	1
ISR	0

Disassembly

```

0x0000029E 2301 MOVS r3,#0x01
0x000002A0 4614 MOV r4,r2
0x000002A2 0014 MOVS r4,r2
0x000002A4 18D5 ADDS r5,r2,r3
0x000002A6 1AD6 SUBS r6,r2,r3
0x000002A8 480D LDR r0,[pc,$52] ; @0x000002E0
0x000002A9 490E LDR r1,[pc,$56] ; @0x000002E4
0x000002A0 2203 MOVS r2,#0x03
0x000002A2 2307 MOVS r3,#0x07
0x000002B0 2400 MOVS r4,#0x00
0x000002B2 2500 MOVS r5,#0x00
0x000002B4 46A8 MOV r8,r5
0x000002B6 46A9 MOV r9,r5
0x000002B8 180E ADDS r6,r1,r0
0x000002B9 D000 BEQ 0x000002BE
0x000002BC 4414 ADD r4,r4,r2
0x000002BD D100 BNE 0x000002C2
0x000002C0 441C ADD r4,r4,r3
0x000002C2 D200 BCS 0x000002C6
0x000002C4 4415 ADD r5,r5,r2
0x000002C6 D300 BCC 0x000002CA
0x000002C8 441D ADD r5,r5,r3
0x000002CA D400 BMI 0x000002CE

```

main.c startup_MKL46Z4.s exp26_asm.s

Registers

Register	Value
Core	
R0	0x7F000000
R1	0x81000000
R2	0xFFFFFFFF
R3	0x00000001
R4	0xFFFFFFF
R5	0x00000000
R6	0xFFFFFFFF
R7	0x00000001
R8	0x00000003
R9	0x00000003
R10	0x000002F8
R11	0x000002F8
R12	0x00000000
R13 (SP)	0xFFFFE150
R14 (LR)	0x0000024B
R15 (PC)	0x000002A2
xPSR	0x21000000
N	0
Z	0
C	1
V	0
T	1
ISR	0

Disassembly

```

0x0000029E 2301 MOVS r3,#0x01
0x000002A0 4614 MOV r4,r2
0x000002A2 0014 MOVS r4,r2
0x000002A4 18D5 ADDS r5,r2,r3
0x000002A6 1AD6 SUBS r6,r2,r3
0x000002A8 480D LDR r0,[pc,#$2] ; @0x000002E0
0x000002AA 490E LDR r1,[pc,#$6] ; @0x000002E4
0x000002AC 2203 MOVS r2,#0x03
0x000002AE 2307 MOVS r3,#0x07
0x000002B0 2400 MOVS r4,#0x00
0x000002B2 2500 MOVS r5,#0x00
0x000002B4 46A8 MOV r8,r5
0x000002B6 46A9 MOV r9,r5
0x000002B8 180E ADDS r6,r1,r0
0x000002BA D000 BEQ 0x000002BE
0x000002BC 4414 ADD r4,r4,r2
0x000002BE D100 BNE 0x000002C2
0x000002C0 441C ADD r4,r4,r3
0x000002C2 D200 BCS 0x000002C6
0x000002C4 4415 ADD r5,r5,r2
0x000002C6 D300 BCC 0x000002CA
0x000002C8 441D ADD r5,r5,r3
0x000002CA D400 BMI 0x000002CE

```

Registers

Register	Value
Core	
R0	0x7F000000
R1	0x81000000
R2	0xFFFFFFFF
R3	0x00000001
R4	0xFFFFFFFF
R5	0x00000000
R6	0xFFFFFFFF
R7	0x00000001
R8	0x00000003
R9	0x00000003
R10	0x000002F8
R11	0x000002F8
R12	0x00000000
R13 (SP)	0xFFFFE150
R14 (LR)	0x0000024B
R15 (PC)	0x000002A4
xPSR	0x21000000
N	0
Z	0
C	1
V	0
T	1
ISR	0

Disassembly

```

0x0000029E 2301 MOVS r3,#0x01
0x000002A0 4614 MOV r4,r2
0x000002A2 0014 MOVS r4,r2
0x000002A4 18D5 ADDS r5,r2,r3
0x000002A6 1AD6 SUBS r6,r2,r3
0x000002A8 480D LDR r0,[pc,#$2] ; @0x000002E0
0x000002AA 490E LDR r1,[pc,#$6] ; @0x000002E4
0x000002AC 2203 MOVS r2,#0x03
0x000002AE 2307 MOVS r3,#0x07
0x000002B0 2400 MOVS r4,#0x00
0x000002B2 2500 MOVS r5,#0x00
0x000002B4 46A8 MOV r8,r5
0x000002B6 46A9 MOV r9,r5
0x000002B8 180E ADDS r6,r1,r0
0x000002BA D000 BEQ 0x000002BE
0x000002BC 4414 ADD r4,r4,r2
0x000002BE D100 BNE 0x000002C2
0x000002C0 441C ADD r4,r4,r3
0x000002C2 D200 BCS 0x000002C6
0x000002C4 4415 ADD r5,r5,r2
0x000002C6 D300 BCC 0x000002CA
0x000002C8 441D ADD r5,r5,r3
0x000002CA D400 BMI 0x000002CE

```

Registers

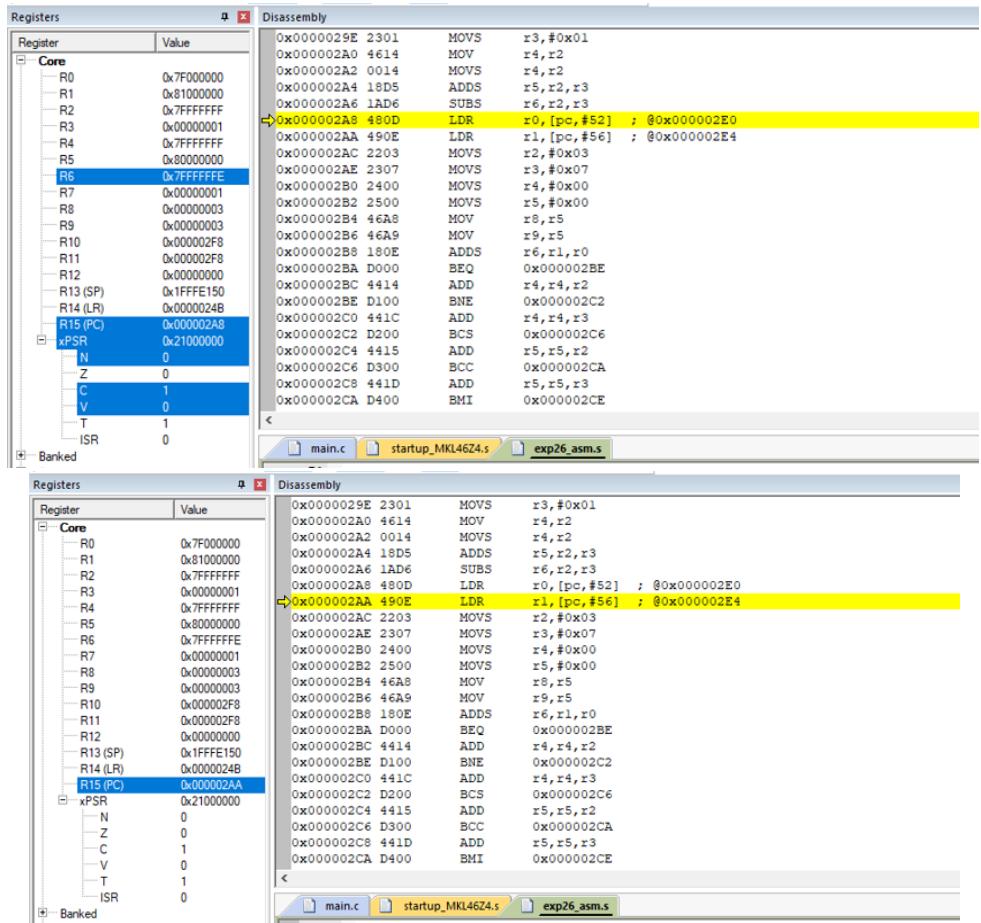
Register	Value
Core	
R0	0x7F000000
R1	0x81000000
R2	0xFFFFFFFF
R3	0x00000001
R4	0xFFFFFFFF
R5	0x80000000
R6	0xFFFFFFFF
R7	0x00000001
R8	0x00000003
R9	0x00000003
R10	0x000002F8
R11	0x000002F8
R12	0x00000000
R13 (SP)	0xFFFFE150
R14 (LR)	0x0000024B
R15 (PC)	0x000002A6
xPSR	0x91000000
N	1
Z	0
C	0
V	1
T	1
ISR	0

Disassembly

```

0x0000029E 2301 MOVS r3,#0x01
0x000002A0 4614 MOV r4,r2
0x000002A2 0014 MOVS r4,r2
0x000002A4 18D5 ADDS r5,r2,r3
0x000002A6 1AD6 SUBS r6,r2,r3
0x000002A8 480D LDR r0,[pc,#$2] ; @0x000002E0
0x000002AA 490E LDR r1,[pc,#$6] ; @0x000002E4
0x000002AC 2203 MOVS r2,#0x03
0x000002AE 2307 MOVS r3,#0x07
0x000002B0 2400 MOVS r4,#0x00
0x000002B2 2500 MOVS r5,#0x00
0x000002B4 46A8 MOV r8,r5
0x000002B6 46A9 MOV r9,r5
0x000002B8 180E ADDS r6,r1,r0
0x000002BA D000 BEQ 0x000002BE
0x000002BC 4414 ADD r4,r4,r2
0x000002BE D100 BNE 0x000002C2
0x000002C0 441C ADD r4,r4,r3
0x000002C2 D200 BCS 0x000002C6
0x000002C4 4415 ADD r5,r5,r2
0x000002C6 D300 BCC 0x000002CA
0x000002C8 441D ADD r5,r5,r3
0x000002CA D400 BMI 0x000002CE

```



Comment/Explanation:

It can be seen that the flags results are different due to the different values. What is more, flags will update after execute each instruction.

For the initial condition, only N flag is set in the final result.

When R₂ is set to 0xFFFFFFFF, N and C flags are set in the final result.

When R₂ is set to 0x7FFFFFFF, only the C flag is set.

15. Answer to Q14 [4 marks]

Answer:

Four conditional branch instructions are executed, while other 4 instructions are not executed.

Explanation:

The execution results are shown in the following:

address	instructions	flags	results
---------	--------------	-------	---------

0x000002BA	BEQ 0x000002BE	Z = 1	execute
0x000002BE	BNE 0x000002C2	Z = 1	Not execute
0x000002C2	BCS 0x000002C6	C = 1	execute
0x000002C6	BCC 0x000002CA	C = 1	Not execute
0x000002CA	BMI 0x000002CE	N = 0	Not execute
0x000002CE	BPL 0x000002D2	N = 0	execute
0x000002D2	BVS 0x000002D6	V = 0	Not execute
0x000002D6	BVC 0x000002DA	V = 0	execute

The program executed according to the instructions and flags. If the codes meet the requirements, the program will jump to the certain address accordingly. Otherwise, it will not change the order.

16. Answer to Q15 [4 marks]

Answer:

Four conditional branch instructions are executed, while other four instructions are not executed.

Explanation:

The execution results are shown in the following table:

address	instructions	flags	results
0x000002BA	BEQ 0x000002BE	Z = 0	Not execute
0x000002BE	BNE 0x000002C2	Z = 0	execute
0x000002C2	BCS 0x000002C6	C = 0	Not execute
0x000002C6	BCC 0x000002CA	C = 0	execute
0x000002CA	BMI 0x000002CE	N = 1	execute
0x000002CE	BPL 0x000002D2	N = 1	Not execute
0x000002D2	BVS 0x000002D6	V = 1	execute

0x0000002D6	BVC 0x0000002DA	V = 1	Not execute
-------------	-----------------	-------	-------------

The program executed according to the instructions and flags. If the codes meet the requirements, the program will jump to the certain address accordingly. Otherwise, it will not change the order.

17. Screenshot of the result of Section 13, Part IV [3 marks]

Screenshots showing that the “mbed_blinky” program has been successfully exported from the Mbed Compiler and opened in Keil uVision (no need to build it).

Screenshot (PrintScreen or Screenshot):

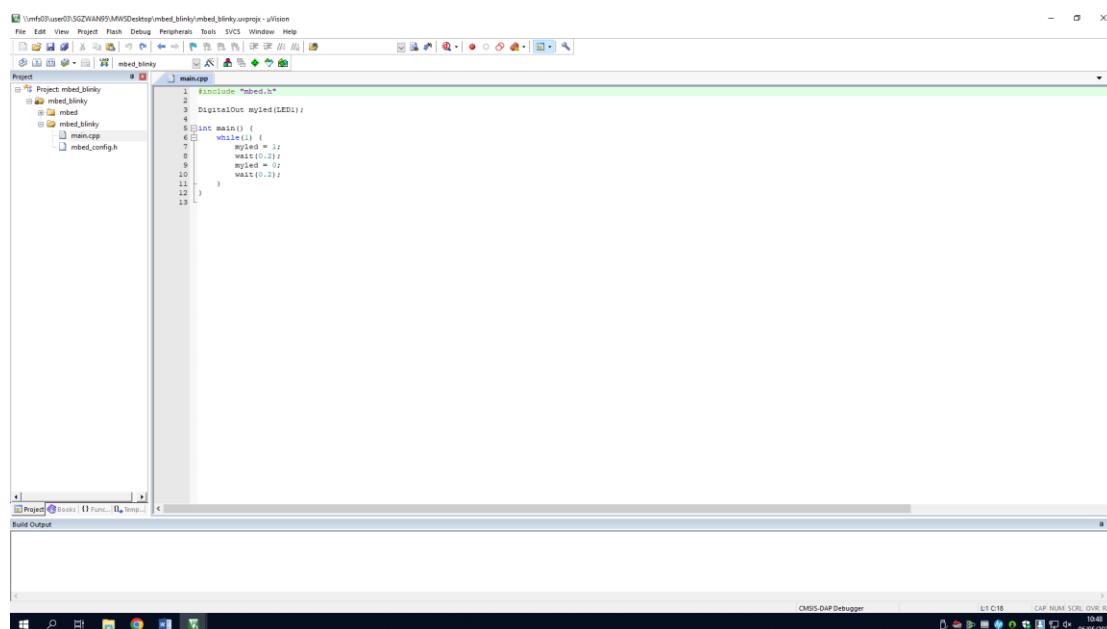


Figure 9

Screenshot (Snipping tool, Preview or equivalent):

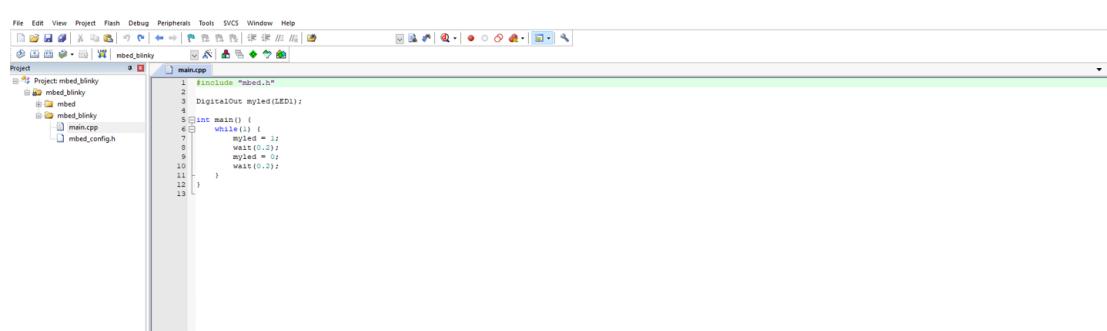


Figure 10

**18. Code of the Hello World programme (Section 12) after editing (put as text
NOT as a screenshot, screenshots of code will receive zero marks) [5 marks]**

Code:

```
#include "mbed.h"

DigitalOut led2(LED2);
DigitalOut led1(LED1);

int main() {
    while(1) {

        led2 = 0;
        led1 = 1;
        wait(0.2);
        led1 = 0;
        led2 = 1;
        wait(0.2);

    }
}

#include "mbed.h"

DigitalOut myled1(LED1);
DigitalOut myled2(LED2);

int main() {
    while(1) {

        myled1 = 1;
        myled2 = 0;
        wait(0.015);
        myled1= 0;
        myled2 = 1;
        wait(0.015);

    }
}
```

Explanation:

For the first version of code, it changes the LED1 to LED2 means change the green LED to red LED instead. Then, the red LED will flash every 0.2 second.

For the second version of the code, we can observe that all the two LEDs are on.

19. Answer to Q16 [4 marks]**Answer:**

The smallest time is 0.015s.

$$\text{The frequency is } \frac{1}{T} = \frac{1}{2*0.015} = 33.33\text{Hz}$$

Explanation:

When the smallest time set to 0.015s, people can not recognise the LED is flashing. They just think the LED is always on.

20. Screenshot of the result of Section 15, Part XI (the output screen) [3 marks]

Screenshot showing the effects of the changes made.

Screenshot (PrintScreen or Screenshot):

```
COM9 - PUTTY
Hello World
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
```

Figure 11: (\n)

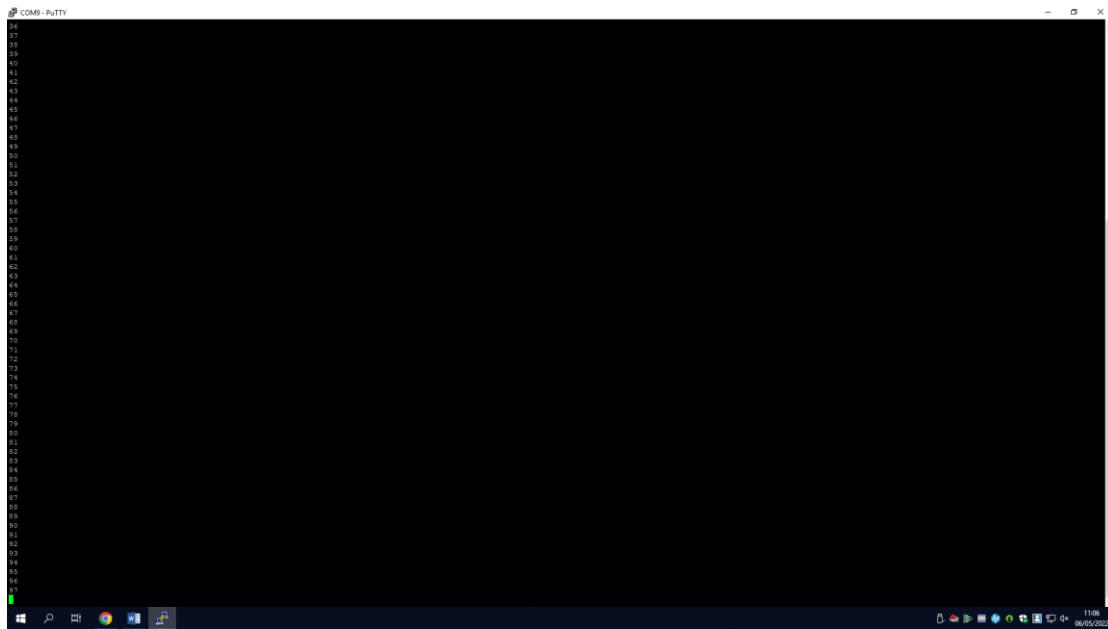


Figure 12: ($\backslash n \backslash r$)

Screenshot (Snipping tool, Preview or equivalent):

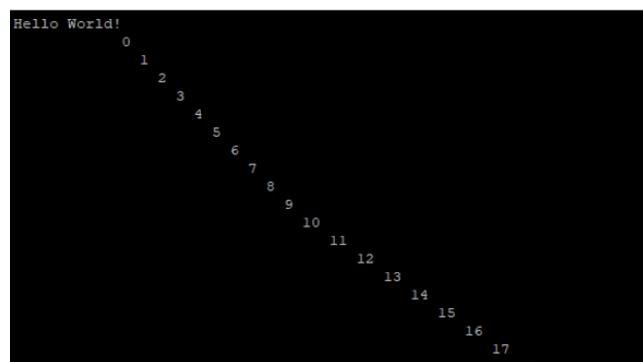


Figure 13

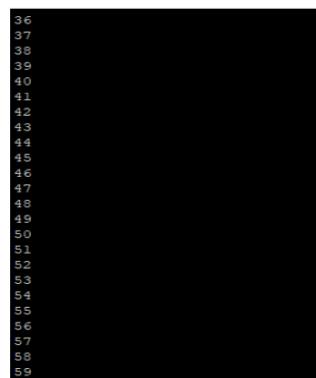


Figure 14

Comment/Explanation:

It can be found that “Hello World” will firstly print on the screen for the both conditions. Then, the program will count number from 0 and increment by 1. For the “\n” condition, it can be seen that each number has different horizontal and vertical position. But for “\n\r” condition, all the numbers are at the left side column. What is more, when speed is set to 115200, the number will print faster than the 9600 condition.

21. The modified code of Section 16 (put as text NOT as a screenshot, screenshots of code will receive zero marks) [5 marks]

Code:

```
#include "mbed.h"
#include "SLCD.h"
#include "tsi_sensor.h"

// Very simple program to read the analog slider and print its value
// on the LCD. Also flashes the RED led.
// -- Al Williams

/* This defines will be replaced by PinNames soon */
#if defined (TARGET_KL25Z) || defined (TARGET_KL46Z)
#define ELEC0 9
#define ELEC1 10
#elif defined (TARGET_KL05Z)
#define ELEC0 9
#define ELEC1 8
#else
#error TARGET NOT DEFINED
#endif

TSIAalogSlider tsi(ELEC0, ELEC1, 40);

DigitalOut gpo(D0);
DigitalOut led1(LED_RED);
DigitalOut led2(LED_GREEN);

SLCD slcd;
int main()
{
    while (true) {
        float f=tsi.readPercentage();
```

```

slcd.printf("%1.3f", f);

if (f==0)
{
    led1 = 1;
    led2 = 1;
}
else if ((0.5>f)&&(f>0))
    led1 =0; // toggle led
    led2=1;

}

else
{
    led2 = 0; // toggle led
    led1=1;

}

}
}

```

Explanation:

When the finger does not touch the area ($f = 0$), no LED is illuminated. When, the finger touches the point between the midpoint and the left-hand ($f < 0.5$), only the red LED is illuminated, while the green LED is illuminated for the other half area.