Antwoorden week 2 – Microprocessorarchitectuur

In dit document vind je de antwoorden van de theorievragen voor week 2 van EMS20. Deze vragen kun je vinden op: https://bitbucket.org/HR_ELEKTRO/ems20/wiki/Theorie/Theorie_Week_2.pdf.

Het is verraderlijk om de onderstaande antwoorden in te zien, zonder eerst zelf de vraag te beantwoorden. Je denkt dan al snel: "inderdaad, dat is het juiste antwoord". Maar als je straks op de toets zelf het antwoord moet bedenken, dan blijkt een en ander toch niet zo eenvoudig te zijn.

Gebruik dit document alleen om je eigen antwoorden te controleren.

2.1

A Fetch (voor het verhogen van de program counter) en execute.

B Fetch

C Execute.

D Store.

E Execute.

F Store.

2.2

A 11000100.

B 10xx0011. $x = don't\ care.$

C 01xx1011.

D 00101010.

2.3 Inhoud 0xFF00FF00 = 0xEF.
Inhoud 0xFF00FF01 = 0xCD.

2.4

A 0x7FFB.

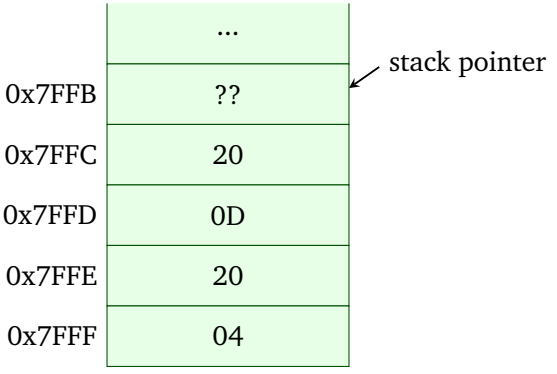
B Zie figuur 1.

C 0x7FFF.

D 0x7FFB.

E Zie figuur 1.

F Het adres dat het laatste op de stack is gezet (Last In), namelijk 0x200D, wordt er als eerste weer afgehaald (Last Out).



Figuur 1: De stack op het moment dat de program counter de waarde 0x2008 bevat.

2.5 **A** Als de stack naar beneden groeit dan is het logisch om zo hoog mogelijk te beginnen. Het overige RAM geheugen bestaat dan uit één stuk.

B Bij het reserveren van RAM-plaatsen voor variabelen is het logisch om vooraan (op het laagste adres van) het RAM te beginnen. Als de stack dan naar beneden groeit (en bovenaan, op het hoogste adres van het RAM wordt geïnitialiseerd), dan is de kans het kleinst dat er een stack overflow optreedt.

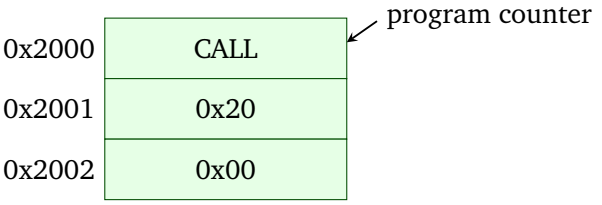
2.6 **A** Er wordt dan data overschreven die pas later in het programma gebruikt zal worden. De plaats in het programma waar de fout zichtbaar wordt is ver verwijderd van de plaats waar de fout optreedt.

B Zie figuur 2.

C Zie listing 1.

2.7 **7.7** 2^{12} bytes.

7.8 R/$\overline{W}$ input. 2^{10} bits.



Figuur 2: Een programma dat (uiteindelijk) altijd een stack overflow zal veroorzaken.

```
int main(void)
{
    return main();
}
```

Listing 1: Een C-programma dat (uiteindelijk) altijd een stack overflow zal veroorzaken.

7.9 2^{10} bits.

7.10 Dat is lastig tekenen, want A en B zitten in elkaar verweven. Zie tabel 1.

7.11 0x71 = 01110001. De even bits $b_6b_4b_2b_0$ komen in module A: 1101. De oneven bits $b_7b_5b_3b_1$ komen in module B: 0100.

Tabel 1: Memory map van het computersysteem gegeven in figuur 7.58 van je boek.

startadres	eindadres	module
0x300	0x3FF	A en B
0x950	0x95F	I/O
0xF00	0xF7F	C
0xF80	0xFFF	D

2.8 **A** 2^{32}.

B 8.

C A_{31}=0, A_{30}=0, A_{29}=0, A_{28}=0, A_{27}=0, A_{26}=0, A_{25}=0, A_{24}=0, A_{23}=1, A_{22}=1, A_{21}=1, A_{20}=1, A_{19}=1, A_{18}=1, A_{17}=1, A_{16}=1, A_{15}=0, A_{14}=0, A_{13}=0, A_{12}=0, A_{11}=0, A_{10}=0, A_{09}=0, A_{08}=0, A_{07}=1, A_{06}=1, A_{05}=1, A_{04}=1, A_{03}=1, BE_7=0, BE_6=0, BE_5=0, BE_4=0, BE_3=0, BE_2=0, BE_1=0, BE_0=1.

D D_7 t/m D_0.

E A_{31}=0, A_{30}=0, A_{29}=0, A_{28}=0, A_{27}=0, A_{26}=0, A_{25}=0, A_{24}=0,
A_{23}=1, A_{22}=1, A_{21}=1, A_{20}=1, A_{19}=1, A_{18}=1, A_{17}=1, A_{16}=1,
A_{15}=0, A_{14}=0, A_{13}=0, A_{12}=0, A_{11}=0, A_{10}=0, A_{09}=0, A_{08}=0,
A_{07}=1, A_{06}=1, A_{05}=1, A_{04}=1, A_{03}=1,
BE_7=0, BE_6=0, BE_5=0, BE_4=0, BE_3=1, BE_2=0, BE_1=0, BE_0=0.

F D_{31} t/m D_{24}.

G A_{31}=0, A_{30}=0, A_{29}=0, A_{28}=0, A_{27}=0, A_{26}=0, A_{25}=0, A_{24}=0,
A_{23}=1, A_{22}=1, A_{21}=1, A_{20}=1, A_{19}=1, A_{18}=1, A_{17}=1, A_{16}=1,
A_{15}=0, A_{14}=0, A_{13}=0, A_{12}=0, A_{11}=0, A_{10}=0, A_{09}=0, A_{08}=0,
A_{07}=1, A_{06}=1, A_{05}=1, A_{04}=1, A_{03}=1,
BE_7=0, BE_6=0, BE_5=1, BE_4=1, BE_3=0, BE_2=0, BE_1=0, BE_0=0.

H D_{47} t/m D_{32}.

I A_{31}=0, A_{30}=0, A_{29}=0, A_{28}=0, A_{27}=0, A_{26}=0, A_{25}=0, A_{24}=0,
A_{23}=1, A_{22}=1, A_{21}=1, A_{20}=1, A_{19}=1, A_{18}=1, A_{17}=1, A_{16}=1,
A_{15}=0, A_{14}=0, A_{13}=0, A_{12}=0, A_{11}=0, A_{10}=0, A_{09}=0, A_{08}=0,
A_{07}=1, A_{06}=1, A_{05}=1, A_{04}=1, A_{03}=1,
BE_7=1, BE_6=1, BE_5=1, BE_4=1, BE_3=0, BE_2=0, BE_1=0, BE_0=0.

J D_{63} t/m D_{32}.

K A_{31}=0, A_{30}=0, A_{29}=0, A_{28}=0, A_{27}=0, A_{26}=0, A_{25}=0, A_{24}=0,
A_{23}=1, A_{22}=1, A_{21}=1, A_{20}=1, A_{19}=1, A_{18}=1, A_{17}=1, A_{16}=1,
A_{15}=0, A_{14}=0, A_{13}=0, A_{12}=0, A_{11}=0, A_{10}=0, A_{09}=0, A_{08}=0,
A_{07}=1, A_{06}=1, A_{05}=1, A_{04}=1, A_{03}=1,
BE_7=1, BE_6=1, BE_5=1, BE_4=1, BE_3=1, BE_2=1, BE_1=1, BE_0=1.

L D_{63} t/m D_0.

M Om iets in één keer in te lezen moeten de adreslijnen A_{31} t/m A_3 gelijk zijn. Met BE_7 t/m BE_0 kunnen we dan een aantal bytes (maximaal acht) in een keer selecteren. Voor het eerste van deze acht bytes (het beginadres van de 8-bytes variabele) geldt dan A_2=0, A_1=0 en A_0=0. Elk adres dat eindigt op 000 (binair) is deelbaar door 8. Dus: als de CPU een 8-bytes getal in één keer in moet kunnen lezen dan moet het beginadres van deze variabele deelbaar zijn door 8.